FRG
German Patent Office
Patent Disclosure Document
DE 196 06 178 A1
Int. Cl.[6]: **H 04 N 11/02** H 04 N 1/64 H 03 M 7/30 G 08 T 9/00
File No.: 196 06 178.4
Filing date: 2/20/96
Date laid open to public inspection: 9/26/96
Union priority: 2/21/95 US 391679
Applicant: Ricoh Co. Ltd., Tokyo, JP
Representative: Sandmair, Marx, 81677 Munich
Inventors: Gormish, Michael J., Menlo Park, Calif. US
Schwarz, Edward L., Menlo Park, Calif. US
Citations:
DE 38 04 175 C2
Jain, Anil K.: Fundamentals of Digital Image Processing: Englewood Cliffs, New Jersey, Prentice-Hall, Inc. 1989, p. 476-483 ISBN 0-13-332578-4
Examination request filed pursuant to § 44 of the Patent Law

Process for compression of an arrangement of pixel values and for decompression of an arrangement of pixel values from a compressed data set

A data compression system divides the input data before compression into color planes. If necessary for game consoles, compression is done for a game cartridge. To minimize the number of passes which the coder/decoder must make, color planes are arranged according to density and the densest color plane is coded first. After the first color plane is coded, the other color planes are coded, but the pixels for which the colors are known from the color planes coded beforehand are not coded. The last color plane is not coded, but it is excluded from all other color planes. Alternatively pixel color values are displayed by vectors with components which have been coded separately by subcolor planes. Likewise each color plane can be coded until a threshold number of pixels is coded, and the remainder have been

coded by one image plane. The video data could be coded using pixel position information as context.

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) Offenlegungsschrift

(10) DE 196 06 178 A1

(51) Int. Cl.[6]:
H 04 N 11/02
H 04 N 1/64
H 03 M 7/30
G 06 T 9/00

(21) Aktenzeichen: 196 06 178.4
(22) Anmeldetag: 20. 2. 96
(43) Offenlegungstag: 26. 9. 96

DE 196 06 178 A1

(30) Unionspriorität: (32) (33) (31)
21.02.95 US 391679

(71) Anmelder:
Ricoh Co., Ltd., Tokio/Tokyo, JP

(74) Vertreter:
Schwabe, Sandmair, Marx, 81677 München

(72) Erfinder:
Gormish, Michael J., Menlo Park, Calif., US;
Schwartz, Edward L., Menlo Park, Calif., US

(56) Entgegenhaltungen:
DE 38 04 175 C2
JAIN, Anil K.: Fundamentals of Digital Image Processing;
Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1989, S. 476-483 ISBN 0-13-332578-4;

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Verfahren zum Komprimieren einer Anordnung von Pixelwerten und zum Dekomprimieren einer Anordnung von Pixelwerten aus einem komprimierten Datensatz

(57) Ein Datenkompressionssystem teilt eingegebene Daten vor einer Kompression in Farbebenen auf. Falls es für Spiel-Konsolen erforderlich ist, wird eine Kompression bei einer Spielkassette durchgeführt. Um die Anzahl an Durchläufen, die ein Codierer/Decodierer durchführen muß, zu minimieren, werden Farbebenen entsprechend der Dichte geordnet und die dichteste Farbebene wird zuerst codiert. Nachdem die erste Farbebene codiert ist, werden die anderen Farbebenen codiert, jedoch Pixel, bei denen aus den vorher codierten Farbebenen die Farben bekannt sind, werden nicht codiert. Die letzte Farbebene wird nicht codiert, sondern sie wird von allen anderen Farbebenen ausgeschlossen. Alternativ hierzu werden Pixel-Farbwerte durch Vektoren mit Komponenten dargestellt, die gesondert durch Unterfarben-Ebenen codiert worden sind. Ebenso kann jede Farbebene codiert werden, bis eine Schwellenwertzahl von Pixels codiert ist und die restlichen durch eine Bitebene codiert worden sind. Die Bilddaten könnten mit Hilfe von Pixel-Positionsinformation als Kontext codiert werden.

DE 196 06 178 A1

Beschreibung

Die Erfindung betrifft das Gebiet der Datenkompression, insbesondere einer Kompression von palettisierten Bilddaten, und betrifft insbesondere ein Verfahren zum Komprimieren einer Anordnung von Pixelwerten und zum Dekomprimieren einer Anordnung von Pixelwerten aus einem komprimierten Datensatz.

Meistens wird bei Bildkompression angenommen, daß eine kontinuierliche Grautonskala oder ein Farbbild komprimiert wird. Bei einem kontinuierlichen Grauton-Bild hat ein kleiner Fehler in dem Wert eines Pixels eine kleine Farbverschiebung dieses Pixels zur Folge. Palettisierte (oder "indexierte") Bilder unterscheiden sich von kontinuierlichen Grautonbildern dadurch, daß ein kleiner Fehler in einem Pixelwert zu einer vollständig anderen Farbe führen kann, und zwar deswegen, da der Farbwert eines Pixel ein Index in einer Farbentabelle (Palette) ist. In der Kopierbranche werden palettisierte Bilder oft als "Spotfarben" Bilder bezeichnet. Offensichtlich sind also die Probleme, palettisierte Bilder verlustfrei hoch zu komprimieren, auch ein Problem in der Kopierbranche.

In Verbindung mit palettisierten Bildern kann im allgemeinen keine Information über die eventuelle Farbe eines Pixels durch den Pixel-Farbwert ohne Bezugnahme auf die Farbentabelle bestimmt werden. Obwohl dies ein Nachteil zu sein scheint, ist es ganz brauchbar, wenn mehr als ein Bild in einem System verwendet wird, wobei der einzige Unterschied die Farbtabelle ist. Beispielsweise kann das negative (inverse) Bild eines palettisierten Bildes dadurch erhalten werden, daß die Farben in der Farbentabelle durch ihre Umkehrwerte ersetzt werden.

Ein Nachteil bei palletisierten Bildern besteht darin, daß sie nicht mittels eines verlustbehafteten Kompressionsprozesses komprimiert werden können, sondern verlustfrei komprimiert werden müssen. Verlustbehaftete Kompressionsprozesse sind im allgemeinen günstig für eine Bildkompression, da sie im allgemeinen eine hohe Kompression bei geringer Leistung schaffen. Ein palettisiertes Bild weist zwei Teile auf, die zweidimensionale Anordnung von Pixel-Farbwerten und die Farbentabelle, die verwendet wird, um die Pixel-Farbwerte in Farben zu übertragen; da jedoch die Farbentabelle viel weniger Speicher als die Anordnung von Pixel-Farbwerten erfordert, wird Kompression oft nur bei der Anordnung von Pixel-Farbwerten in Verbindung mit der Farbentabelle angewendet, in der Nicht-Komprimiertes zusammen mit dem komprimierten Bild enthalten ist.

Die am 1. Dezember 1994 eingereichte US-Patentanmeldung S.N. 08/347 789 der Anmelderin geht auf eine Erfindung von Michael Gormish und Martin Boliek und hat den Titel "Datenkompression für palettisierte Videobilder". (Diese Anmeldung wird nachstehend als die "Gormish/Boliek-Anmeldung" bezeichnet.) Die Gormish/Boliek-Anmeldung zeigt ein Verfahren, um wirksam palettisierte Bilder mit hohen Kompressionsraten zu komprimieren. Hochkomprimierte palettisierte Bilder werden bei vielen Anwendungen benötigt, wie bei Spielkassetten, welche eine Speicherung für die Bilder erfordern, die in einem Spiel (Video-Spiel, einem in der Hand gehaltenen Spieler, einem Computer-Spiel, usw.) verwendet werden. Hochkomprimierte, palettisierte Bilder sind auch erwünscht in Verbindung mit Online-Diensten und dem "Informatifons-Superhighway". Das weltweite Netz (World Wide Web-WWW) wird immer

beliebter, um Informat[illegible]er das Internet zu verbreiten. Zum Teil beruht die Beliebtheit des WWW auf der Tatsache, daß die Grundeinheiten des WWW HTML-(Hyper Text Mark Up Language-)Dokumente sind, welche eingebettete Grafiken mit Text enthalten können. Diese Dokumente enthalten oft Bilder, welche palettisiert werden. Viele Benutzer des weltweiten Netzes (WWW) haben Zugriff auf diese Dokumente mit Hilfe eines Personal-Computers, wobei sie über ein Modem über normale Fernsprechleitungen mit dem Internet verbunden sind. Bei dieser Konfiguration ist ein Benutzer auf Datenraten von 14 400 Bits/s oder 28 800 Bits/s beschränkt.

Somit ist für einen rechtzeitigen Empfang von komplexen palettisierten Bildern die Fähigkeit, diese Bilder für eine Übertragung hoch zu komprimieren, der Schlüssel für die zunehmende Popularität des WWW. Obwohl schnellere Modems entwickelt sind, gibt es bereits eine Forderung nach komplexeren Grafiken. Folglich wird eine Kompression in absehbarer Zukunft notwendig.

Viele in dem WWW verwendeten Bilder werden als grafische Austauschformat-(GIF-)Dateien gespeichert und übertragen, bei welchen eine Ziv-Lempel-Kompression benutzt wird. Die Ziv-Lempel-Kompression behandelt das Bild als eine eindimensionale Folge von Indexwerten, welche die zweidimensionale Art von Bilddaten ignoriert.

Was folglich benötigt wird, ist eine Kompressionseinrichtung und ein -Verfahren, welche der Art nach palettisierte Bilddaten benutzen, um die Bilddaten hoch zu komprimieren; ferner wird ein entsprechender Dekompressor benötigt.

Durch die Erfindung ist eine verbesserte Kompression von palettisierten Bilddaten geschaffen, indem die Bilddaten bei Farbe (für zweidimensionale Bilder wird dies als ein "Codieren in der Farbebene" bezeichnet) komprimiert (codiert) werden. Eine Farbebene für ein nicht-binäres Bild ist ein binäres Bild, in welchem jeder Pixel-Farbwert des nicht-binären Bildes durch ein Bit ersetzt wird, das anzeigt, ob dieses Pixel "in" dieser Farbebene ist oder nicht. Ein Pixel ist in einer Farbebene, wenn der Pixel-Farbwert gleich einem Farbwert ist, welcher der Farbebene zugeordnet ist.

Gemäß der Erfindung codiert ein Kompressor eine erste Farbebene, codiert dann eine zweite Farbebene und fährt mit dem Codieren für jede folgende Farbebene fort, bis alle Farbebenen, die Pixel in sich aufweisen, codiert sind, außer bei der "letzten" Farbebene. Beim Codieren jeder Farbebene werden Pixel, welche in vorherigen Farbebenen liegen, ignoriert, da ein Pixel nur in einer Farbebene sein kann. Da jede codierte Farbebene eine binäre Anordnung ist, kann sie mittels eines binären Bildcodierers codiert werden.

Beim Decodieren werden die Farbebenen in derselben Reihenfolge decodiert. Die erste Farbebene wird decodiert und zeigt an, welche Pixel sich in der ersten Farbebene befinden; dann wird die zweite Farbebene decodiert, um anzuzeigen, welche Pixel sich in der zweiten Farbebene befinden, und welche in den nachfolgenden Farbebenen sind. (Die codierte zweite Farbebene gibt weder einen Hinweis darauf, welche Pixel sich in der ersten Farbebene befinden, noch braucht sie dies zu tun.) Sobald alle, aber zumindest eine codierte Farbebene decodiert worden ist, folgert der Dekompressor, daß irgendwelche Restpixel von der letzten Farbe sind.

Natürlich erfordert ein Farbebenen-Codieren von nicht-binären Bildern mehr Durchgänge als ein anderes

Codieren, wie ein Bitebenen-Co[illegible]. Beispielsweise wird ein 8 Bit Farbbild in acht (8) Durchgängen mit Hilfe des Bitebenen-Codierens codiert, aber es können bis 255 Durchgänge für ein vorgegebenes Pixel bei einem Farbebenen-Codieren erforderlich sein. Um die Anzahl an geforderten Durchgängen zu minimieren, werden die Farbebenen nach Dichte geordnet, und die dichteste Farbebene wird zuerst codiert, wobei sich Dichte auf die Dichte von Pixel in einer Farbebene bezüglich der Gesamtanzahl an Pixel in dem Bild bezieht. Wie vorstehend erläutert, wird ein Pixel codiert, indem es durch eine binäre Anzeige dargestellt wird, ob das Pixel sich in der zu codierenden Farbebene befindet oder nicht. Nachdem die erste Farbebene codiert ist, werden Pixel, bei denen bekannt ist, daß sie Farben von vorherigen codierten Farbebenen haben, nicht mehr codiert. Dadurch daß die dichtesten Farbebenen zuerst codiert werden, wird die Gesamtanzahl an Durchgängen verringert.

In einer weiteren Ausführungsform wird jeder Pixelfarbwert durch einen Vektor dargestellt, und die Komponenten der Vektoren werden gesondert durch "Unterfarben"-Ebenen codiert, wobei jede Unterfarben-Ebene alle die Pixel enthält, welche einen Vektor-Komponentenwert haben, welcher gleich dem Wert für die Unterfarbenebene ist. Auf diese Weise kann ein Bild ohne weiteres parallel komprimiert werden.

In einer weiteren Ausführungsform wird jede Farbebene codiert, bis eine Schwellenwertzahl von codierten Pixel erreicht ist, und dann werden die verbleibenden Pixel durch eine Bitebene codiert.

Ein Entropie-Codierer verwendet Kontext-Information, um eine Kompression eines Pixels zu verbessern, indem der verwendete Code optimiert wird, um das Pixel basierend auf der Wahrscheinlichkeit dieses Pixels, zu codieren, das entsprechend dem Pixel-Kontext vorkommt. Üblicherweise kann in einem Bild der Kontext eines Pixel zum Teil durch dessen benachbarte Pixel bestimmt werden. Wenn die Bilddaten eine Ansammlung von kleineren Bildern, wie Schemen (sprites), Zeichen oder ein grafisches Menü sind, können einige benachbarte Pixel unabhängig von dem zu codierenden Pixel sein, so daß die Kontext-Information die Pixel-Positions-Information enthält, um eine Kompression weiter zu verbessern.

Da ein Codierer mehr Durchgänge bei dem Farbebenen-Codieren als bei anderen Codierschemen erfordert, wird ein schneller Entropie-Codierer verwendet. Ein solches schnelles Codiersystem besteht darin, daß gemäß den Ausführungen in dem US-Patent 5,272,478 der Anmelderin der vorliegenden Erfindung ein B-Codierer als der Bitgenerator für den vorerwähnten Kompressionsprozeß verwendet wird. Eine Alternative hierzu ist der Q-Codierer, der von IBM, New York entwickelt worden ist oder der hochschnelle binäre Entropie-Codierer, der in dem US-Patent 5,381,145 der Anmelderin (Erfinder Allen et. al) offenbart ist.

Die Erfindung schafft auch eine neue Einrichtung zum Durchführen von parallelem Codieren und Decodieren. In einer Ausführungsform wird ein Bild in Bänder aufgeteilt, und zwar eines pro Codierer. Jeder Codierer arbeitet unabhängig von den anderen. Erforderlichenfalls werden fest vorgegebene Kontext-Pixel substituiert, wobei Codierer sich sonst auf die Werte von Pixel außerhalb dieses Bandes beziehen. Die Bänder können dann unabhängig parallel decodiert werden.

In einer anderen Ausführungsform durchlaufen mehrere Codierer das gesamte Bildcodieren für spezielle Farbebenen. Erforderlichenfalls wird das Bild in einem Speicher gespeichert, [illegible]cher gleichzeitig auf leere Speicher Zugriff haben kann, um einen Verlust an Parallelverarbeitung zu vermeiden. In dieser Ausführungsform durchlaufen Codierer das Bild mit variablen Daten in Abhängigkeit von den Daten, auf die sie treffen (Farbebenen niedriger Dichte werden schneller verarbeitet als Farbebenen hoher Dichte). Die Codierer pausieren erforderlichenfalls, um mehrfach Zugriff auf denselben Speicher zu vermeiden, oder wenn ein Pixel für eine vorgegebene Farbebene Kontext von Pixel erfordert, die noch nicht bereits codiert worden sind.

In noch einer weiteren Ausführungsform durchlaufen mehr Codierer das ganze Bild, sind aber nicht frei, um mit irgendeiner Geschwindigkeit zu arbeiten. Beispielsweise können die Codierer bei einem fest vorgegebenen Versatz (Offset) oder in einem begrenzten Fenster gehalten werden, welches sicherstellt, daß zwei Codierer nicht gleichzeitig Zugriff auf dasselbe Pixel haben, und daß kein Codierer Zugriff auf ein Pixel hat, bei welchem Kontext-Pixel nicht schon codiert worden sind.

Die Erfindung ist auch bei der Kompression von nicht-bildbezogene Daten anwendbar, wobei ähnliche Charakteristika der zu verdichtenden Daten bestehen und entsprechende Notwendigkeiten für eine Kompression vorhanden sind.

Nachfolgend wird die Erfindung anhand von bevorzugten Ausführungsformen unter Bezugnahme auf die anliegenden Zeichnungen im einzelnen erläutert. Es zeigen:

Fig. 1 ein schematisches Diagramm, bei welchem die Aufteilung eines Bildes auf Farbebenen gezeigt ist;

Fig. 2 ein Blockdiagramm eines Kompressors und Dekompressors gemäß der Erfindung;

Fig. 3 ein schematisches Diagramm einer Kontext-Modellschablone;

Fig. 4 ein schematisches Diagramm eines Kontext-Modells;

Fig. 5 ein Ablaufdiagramm eines Prozesses gemäß der Erfindung zum Codieren von Farbebenen in einem Kompressionssystem;

Fig. 6 ein Blockdiagramm eines Parallel-Codierers für ein Farbebenen-Codieren;

Fig. 7 ein Blockdiagramm eines Parallel-Codier-Systems;

Fig. 8 ein detaillierteres Blockdiagramm des in **Fig.** 7 dargestellten Parallel-Codiersystems, bei welchem nur ein Codierer dargestellt ist;

Fig. 9 ein Blockdiagramm eines Parallel-Codiersystems, in welchem das Fortschreiten von Codierern mit dem Fortschreiten anderer Codierer verbunden ist;

Fig. 10 eine Tabelle, die zeigt, wie die Codierer in dem Parallel-Codiersystem der **Fig.** 9 sich über die Pixel eines Bildes fortbewegen, bei welchem die Codierer gezwungen sind, mit aufeinanderfolgen Pixeln zu arbeiten, und die Codierer sich nicht fortbewegen, bis das hintere Pixel codiert ist;

Fig. 11 eine Tabelle, die zeigt, wie die Codierer in dem Parallel-Codiersystem der **Fig.** 11 sich über die Pixel eines Bildes fortbewegen, bei welchem Codierer gezwungen sind, so daß, sobald ein Codieren an einem Pixel beginnt, ein Codieren beendet wird, bevor ein neues Pixel codiert wird, und

Fig. 12 eine Tabelle, die zeigt, wie die Codierer in dem Parallel-Codiersystem der **Fig.** 9 sich über Pixel eines Bildes fortbewegen können, bei welchem die Codierer gezwungen sind, so daß ein Codieren an einem neuen Pixel begonnen wird, wenn ein vorderer Codierer verfügbar ist.

Fig. 1 ist ein schematisches Diag[illegible], das die Aufteilung eines Bildes auf Farbebenen anzeigt. Wie nachstehend noch erläutert wird, wird, obwohl ein Bild, das auf Farbebenen aufgeteilt ist, größer als ein nicht-aufgeteiltes Bild ist (für eine 8 Bit-Farbe ist der Unterschied 256 Bit/Pixel gegenüber 8 Bit/Pixel), das aufgeteilte Bild üblicherweise auf eine kleinere Größe komprimierbar ist als das nicht aufgeteilte Bild.

Ein digitales Bild wird durch eine zweidimensionale Anordnung von Pixel dargestellt, wobei jedes Pixel eine Stelle in einem Bildgitter und jedes Pixel einen Farbwert hat. Zur Verdeutlichung zeigt Fig. 1 nur ein (3 × 3) Bild in 16 möglichen Pixelfarben, während in der Praxis Bilder viel mehr Pixel in viel mehr möglichen Farben enthalten. Beispielsweise ist in einer speziellen Ausführungsform ein Bild durch eine Anordnung aus (1024 × 1024) Pixel gebildet, und jedes Pixel hat einen Farbwert, der aus einer Palette von 16,8 Millionen Farben (eine 24Bit Farbe) ausgewählt ist.

Fig. 1(A) zeigt jedes Pixel mit einer Bezeichnung, die dessen Position in dem Bild anzeigt, wobei die erste Ziffer die Zeile für das Pixel und die zweite Ziffer dessen Spalte anzeigt.

Fig. 1(B) gibt den Farbwert für jedes Pixel an. Der Farbwert ist ein ganzzahliger Wert von 0 bis 15, obwohl nicht alle Farbwerte sich eignen, um in dem Bild von Fig. 1 zu erscheinen. Die Farbe 7 ist der häufigste Farbwert, da fünf der Pixel die Farbe 7 haben.

Fig. 1(C) zeigt das Bild, das in gesonderten Farbebenen aufgeteilt ist, und zeigt einen binären Bitstrom, welcher die aufgeteilten Farbebenen darstellt. Jedes Pixel einer gesonderten Farbebene hat einen von drei Werten: "1" zeigt an, daß die Farbe des Pixels zu der Farbe der Farbebene paßt; "0" zeigt an, daß die Farbe des Pixels nicht zu der Farbe der Bildebene paßt und daß die Farbebene, zu der das Pixel paßt, nicht bereits codiert worden ist. "--" zeigt an, daß die Farbe des Pixels zu einer Farbe einer Farbebene paßt, welche bereits codiert worden ist. Die Farbe von Pixel, die mit "--" sind, ist bereits decodiert worden, wenn ein Decodierer auf sie trifft, so daß keine weitere Information über dieses Pixel benötigt wird. Daher braucht dieses Pixel nicht mehr codiert zu werden, und der sich ergebende binäre Strom enthält nur die Einsen ("1") und Nullen ("0") von den aufgeteilten Farbebenen. Wenn nicht die Reihenfolge der Farbebenen festgelegt ist, ist eine Tabelle der Reihenfolge von Farbebenen (die üblicherweise nach der Häufigkeit geordnet werden, daher die Bezeichnung "FT" für die Häufigkeits-(frequency)Tabelle) in dem Datenstrom enthalten.

In einer typischen Ausführungsform ist das vorübergehende Speichern begrenzt, so daß die Farbebenen-Aufteilung nicht eine physische Aufteilung, sondern ein logische ist. Folglich wird die Häufigkeitstabelle (FT) erzeugt und verwendet, um festzulegen, daß die Farbebene 7 zuerst codiert werden sollte. Da jedes Pixel aus dem Bild gelesen wird, wird "1" abgegeben, wenn es ein Pixel von Farbe 7 ist; anderenfalls wird "0" abgegeben. Für nachfolgende Farbebenen wird "1" für Pixel in dieser Farbebene abgegeben, eine "0" wird für Pixel in Farbebenen abgegeben, welche auf die aktuelle Farbebene in der Häufigkeitstabelle folgen, und es wird nichts abgegeben (was durch "--" in Fig. 1(C) bezeichnet ist), für Pixel in Farbebenen, welche vor der aktuellen Farbebene in der Häufigkeitstabelle kommen, wenn diese Pixel bereits voll codiert worden sind. Aufgrund einer willkürlichen Übereinkunft werden Farbebenen mit derselben Dichte zuerst mit der niedrigsten Farbzahl eingeordnet. Folglich [illegible]t in Fig. 1 die Farbebene 3 vor einer (nicht dargestellten) Farbebene 4. Da die Farbebene 4 die letzte Farbebene ist, braucht sie nicht codiert zu werden. Die Farbebenen für nicht-benutzten Farben (z. B. Farbebenen 0, 1, 2, 5, 6, 8, 9, 10, 11, 13, 14 und 15 in Fig. 1) brauchen daher auch nicht codiert zu werden.

Ein Dekompressor erhält den binären Bitstrom (nachdem er codiert ist) und muß das Bild aus der Information in dem Bitstrom wieder zusammensetzen. Wenn der Bitstrom von Fig. 1(C) in einem Codierer vorgesehen wird und der Codierer und der Decodierer sich verständigen, daß die Bildgröße (3 × 3) ist, legt der Decodierer fest, daß die häufigste Farbebene die Farbebene 7 ist, und daß die ersten neun Bits in dem Bitstrom nach der Häufigkeitstabelle (FT) zu dieser Bitebene gehören. Da die Bitebene 7 vier "0" Bits hat, bestimmt der Decodierer auch, daß die nächste Farbebene durch die nächsten vier Bits des Bitstroms codiert wird. Der Codierer kann dann aus der Häufigkeitstabelle (FT) festlegen, daß diese vier Bits für eine Farbebene 12 sind, daß die nächste Farbebene eine Farbebene 3 ist, daß die Farbebene 3 bei den nächsten zwei Bits des Farbstroms codiert wird (da die Farbe 12 zwei "0" Bits hat), und daß die letzte Farbebene die Farbebene 4 ist. Hieraus folgt dann, daß alle restlichen Farbebenen leer sind.

Selbstverständlich kann der vorstehend beschriebene Prozeß auch dazu verwendet werden, um Daten zu komprimieren, welche nicht Bilddaten sind oder welche nicht Farben einschließen, indem erkannt wird, daß jedes Element (wie beispielsweise ein Byte) eines zu komprimierenden Datenblocks analog einem Pixel-Farbwert ist, und eine "Element" -Ebene Elemente gruppiert, die Werte haben, welche gleich dem Wert sind, welcher der Element-Ebene in der Weise zugeordnet ist, daß Pixel gleicher Farbe in Farbebenen gruppiert werden.

Fig. 2 zeigt einen Kompressor 202 und einen Dekompressor 204, welche verwendet werden können, um den gerade beschriebenen Prozeß durchzuführen. Der Kompressor 202 liest nicht-komprimierte Bilddaten (Pixel-Farbwerte) aus einer Datei 206 und setzt die Daten in komprimierte Bilddaten um, welche, wie dargestellt, als Datei 208 gespeichert werden. Diese Datei 208 wird von dem Dekompressor 204 benutzt, um das Originalbild in rekonstruierte Bilddaten umzugestalten, die in einer Datei 210 gespeichert sind. Wenn eine verlustfreie Kompression verwendet wird, ist die Daten 210 eine exakte Kopie der Datei 206.

Der in Fig. 2 dargestellte Kompressor 202 weist einen Ebenen-Separator 212, eine Kontext-Modelleinheit 214 und einen Entropie-Codierer 216 auf. Der in Fig. 2 dargestellte Dekompressor 204 weist einen Entropie-Decodierer 218, eine Kontext-Modelleinheit 220 und einen Ebenen-Akkumulator 222 auf. Die Dateien 206, 208, 210 sind Dateien auf einem Plattenspeicher oder sind Speicherblöcke in einem Computer-Speicher.

In dem Kompressor 202 hat der Ebenen-Separator 212 einen Eingang, um Daten aus der Datei 206 zu lesen und einen Ausgang für separierte Bildebenen-Daten (Wert "1", "0" oder "--"; Pixelstelle, usw.). Eine Kontext-Modelleinheit 214 hat einen Eingang für separierte Bildebenen-Daten und zwei Ausgänge, von denen an einem ein Ergebnis und an dem anderen ein Kontext für dieses Ergebnis abgegeben werden. Der Entropie-Codierer 216 hat Eingänge für das Ergebnis und dessen Kontext und hat einen Ausgang für einen komprimierten Bitstrom, welcher in der Datei 208 gespeichert ist.

In dem Dekompressor 204 hat der Entropie-Decodie-

rer 218 einen Eingang für den komprimierten Bitstrom, einen Eingang für einen Kontext und einen Ausgang zum Abgeben eines Ergebnisses. Die Kontext-Modelleinheit 220 hat einen Eingang, um das Ergebnis von dem Entropie-Decodierer 218 aufzunehmen, einen Ausgang, um einen Kontext an dem Entropie-Decodierer 218 zu schaffen und einen Ausgang, um einen Farbebenen-Bitstrom zu schaffen. Der Ebenen-Akkumulator 222 hat einen Eingang, um den Farbebenen-Bitstrom von der Kontext-Modelleinheit 220 anzunehmen und einen Ausgang zum Abgeben von Bildern an die Datei 210.

In der WWW-(World Wide Web)Umgebung wird die Datei 206 vorher in die Datei 208 komprimiert, und die Datei 208 ist mit einem WWW-Server versehen und wird folglich für Nutzer von WWW-Browser-Kunden verfügbar gemacht. Diese WWW-Kunden würden dann einen Dekompressor 204 haben. Einige Browser-Kunden speichern die nicht-komprimierten Bilddaten in einer Datei, wie beispielsweise der Datei 210, während andere Browser-Kunden das Bild für eine Anzeige dekomprimieren, ohne es jedoch ständig in einer Datei zu speichern, wenn nicht der Browser-Nutzer das sicherzustellende Bit anfordert.

Der Ebenen-Separator 212 setzt ein Bild in einen Bitstrom um, welcher ähnlich dem in Fig. 1(C) dargestellten Bitstrom ist. Dieser Bitstrom wird dann in die Kontext-Modelleinheit 214 eingegeben, welche zusammen mit dem Entropie-Codierer 216 den Bitstrom codiert. Die Kontext-Modelleinheit 214 schafft einen Kontext für jedes Ergebnis, das dem Entropie-Codierer 216 durchlaufen hat, und dieser Kontext ist entweder ein optimierter Gesamt-Kontext oder kann in Abhängigkeit von den Bildern variieren. Der Ebenen-Separator 212 gibt auch die Farb-Häufigkeitstabelle (FT) an die Datei 208 ab. Natürlich braucht, wenn die Reihenfolge von Farben im vorhinein festgelegt ist, die Häufigkeitstabelle (FT) nicht für jede Datei 208 von dem Kompressor 202 zu dem Dekompressor 204 durchlaufen. Erforderlichenfalls ist die Farbtabelle, welche die Pixel-Farbwerte in Farben abbildet, auch in der Datei 208 enthalten.

Da der Dekompressor 204 in der Lage sein muß, das Bild aus dem codierten (komprimierten) Bild zu rekonstruieren, können Pixel, welche vorher codiert wurden, als Kontext-Pixel für ein aktuelles Pixel verwendet werden. Bekanntlich schaffen Pixel in der Nähe des aktuellen Pixels einen brauchbaren Kontext, wenn das aktuelle Pixel codiert wird. Bei anderen Codier-Schemata, bei welchen ein Codierer nur einen Durchlauf über das Bild von oben nach unten und von links nach rechts macht, können die Pixel unter und die Pixel rechts von dem aktuellen Pixel keinen Kontext für das aktuelle Pixel schaffen, da diese Pixel dem Dekompressor unbekannt sind, wenn das aktuelle Pixel zu decodieren ist. Gemäß der Erfindung hat jedoch nach der ersten Farbebene der Decodierer Information über die Pixel darunter und die Pixel rechts davon, nämlich, ob diese Pixel auf den vorher decodierten Bildebenen vorhanden waren oder nicht.

Fig. 3 und 4 veranschaulichen, wie ein Kontext gebildet wird. Fig. 3 zeigt eine Kontext-Modellschablone (template) 300. Eine Kontext-Modellschablone dient dazu, Beziehungen zwischen einem aktuellen Pixel und potentiellen Kontext-Pixel zu bezeichnen, die bei dem Codieren/Decodieren des aktuellen Pixel verwendet werden. In der Kontext-Modellschablone 300 ist eine Stelle des aktuellen Pixels mit "X" bezeichnet, und die umliegenden Pixel-Stellen sind mit T0 bis T23 bezeichnet. In der Praxis verwendet ein Kontextmodell üblicherweise weniger als alle Kontextpixel, die in einer Schablone mit der Größe der Schablone 300 verfügbar sind. Beispielsweise kann ein nicht allzu komplexer Codierer die Farbwerte der Kontextpixel T0 und T2 verwenden. Wenn die Farbwerte durch acht Bits dargestellt werden, dann sind 65 536 $((2_k)^2)$ verschiedene Kontexte für jeden von 256 möglichen aktuellen Pixel-Farbwerten möglich. Wenn nur ein Bit verwendet wird, das anzeigt, ob das Kontext-Pixel in der laufenden bzw. aktuellen Farbebene liegt, kann der Kontext durch mehr Pixel geschaffen werden.

Fig. 4 zeigt ein Kontextmodell 400, welches den Kontext anzeigt, der für einen Codierer verwendet worden ist. Das Kontextmodell 400 basiert auf der Kontext-Modellschablone 300. Die Gormish/Boliek-Anmeldung lehrt, wie ein Codiersystem auszulegen ist, wenn das Kontext-Modell eine sich dynamisch ändernde Funktion der Kontext-Modellschablone ist. Mit dem Kontextmodell 400 können neun Kontext-Pixel verwendet werden, da jedes Pixel nur ein Kontextbit erfordert, nämlich ein Bit, das anzeigt, ob das Kontextpixel in der laufenden Farbebene ist oder nicht. In Fig. 4 ist das laufende bzw. aktuelle Pixel, für welches ein Kontext benötigt wird, mit "X" bezeichnet und seine Kontextpixel sind mit C0 mit C8 bezeichnet.

Als Alternative hierzu könnte der Kontext, welcher durch ein Kontextpixel geschaffen ist, ein Bit sein, das anzeigt, ob das Pixel codiert/decodiert worden ist oder nicht oder der Kontext könnte einer von drei Werten sein, der anzeigt, daß das Kontextpixel in einer vorher codierten Farbebene, der laufenden Farbebene oder in einer noch nicht-codierten Farbebene liegt. Manches von dem Kontext kann auch durch Pixel rechts von dem laufenden Pixel oder unter dem laufenden Pixel geschaffen werden, wie beispielsweise C9 bis C11. Für diese Pixel zeigt das Kontextbit an, ob diese Pixel in einer vorher codierten Farbebene liegen oder nicht. Für nicht-parallele Codiersysteme (und einige Parallel-Codiersysteme) sind die Kontextbits für Pixel C9 bis C11 bereits zu der Zeit bekannt, während welcher das laufende Pixel zu decodieren ist.

In Abhängigkeit von den Bilddaten könnte eine Kontext-Modellmaske, wie diejenige, die in der Gormish/Boliek-Anmeldung dargestellt ist, verwendet werden, um ein Kontextmodell 400 dynamisch zu modifizieren, um auszuwählen, welche Bits von dem Pixel in der Kontext-Modellschablone 300 den Kontext beeinflussen. Erforderlichenfalls könnte Information darüber, wie das Kontextmodell 400 dynamisch modifiziert wird, als Teil in der Kartei 208 erhalten sein. Ebenso könnte, wenn eine darunter liegende Struktur in den Bilddaten vorhanden ist, und das Bild aus unabhängigen Unterbildern gebildet ist, beispielsweise aus Blöcken von (8 × 8) Pixel, die Zeichen oder Schemen (sprites) festlegen, eine Pixel-Position in einem Zeichen oder einem Schemen als zusätzliche Kontext-Information verwendet werden. Wenn beispielsweise ein Bit anzeigt, ob das laufende Pixel in der ersten Zeile eines Zeichens oder innen liegt oder nicht, könnte dieses Bit wertvolle Kontextinformation sein, wenn die Kontextpixel über einem Pixel in einer ersten Zeile Pixel von verschiedenen Zeichen oder Schemen sind und im Vergleich mit Pixel in dem Zeichen oder Schemen nicht mehr in Beziehung sein wollen.

In Fig. 2 wird ein Decodieren in einer Weise durchgeführt, die dem Decodierprozeß entgegengesetzt ist. Der Ebenen-Akkumulator 222 nimmt den Bitstrom von der Kontext-Modelleinheit 220 an und füllt einen internen

Puffer auf, wobei mit den Pixel b[illegible]en wird, welche die häufigste Farbe haben. Wenn alle Pixel erhalten und decodiert sind, enthält der interne Puffer das Bild, welches in die Datei 210 ausgelesen werden kann. Alternativ hierzu können die Pixel abgegeben werden, da sie in der Datei 210 erhalten werden, was einfach durchzuführen ist, da die Datei 210 direkt zugänglich ist. Wenn ein interner Puffer verwendet wird, kann er vorher mit Pixel geladen werden, die alle auf den Farbwert der Pixelebene eingestellt sind, von der bekannt ist, daß sie die letzte auftretende Farbebene ist. Wenn der Puffer vorher mit der Farbe der zuletzt vorkommenden Farbebene geladen ist, würde dies als ein Hinweis/Flag dienen, durch das angezeigt wird, daß das Pixel noch nicht decodiert worden ist.

Fig. 5 ist ein Ablaufdiagramm eines Prozesses zum Codieren von Pixel gemäß der einen Ausführungsform der Erfindung. Wie aus der detaillierten Beschreibung zu ersehen ist, gelten die Arbeitsweise und die Kenndaten für einen Codierer sowie auch für einen Decodierer. Ein Codierer setzt einen ersten Bitstrom basierend auf einem laufenden, zu codierenden Bit und basierend auf Information über vorher codierte Bits in einen zweiten Bitstrom um, wobei der zweite Bitstrom vorzugsweise weniger Bits enthält als der erste Bitstrom. Ein Decodierer setzt den zweiten Bitstrom in den ersten Bitstrom um, indem er dieselben Aktionen wie der Codierer durchführt, um so festzulegen, welches aktuelle Bit bzw. welche aktuellen Bits des ersten Stroms das aktuelle Bit bzw. die aktuellen Bits des zweiten Bitstroms erzeugt haben würden, wenn die Information über vorher codierte Bits vorgegeben ist, (welche der Codierer zur Verfügung hat, da der Decodierer diese Bits bereits decodiert hat). Daher ist ein Codierer äquivalent einem Codierer, solange der Codierer nicht auf Information wartet, welche nicht schon bereits codiert worden ist oder nicht bereits codiert worden sein kann, wenn das aktuelle Bit zu decodieren ist. Folglich wird im Hinblick auf die Verständlichkeit der Begriff "Codierer" sowohl in Verbindung mit einem Codierer in einem Kompressor als auch in Verbindung mit einem Decodierer in einem Dekompressor verwendet.

In Fig. 5 beginnt der Codierprozeß durch Initialisieren eines Zeigers C an der ersten Bitebene (Schritt S1) durch Initialisieren einer Archiv-(history)Anordnung derselben Größe wie das Bild. Die Archiv-Anordnung speichert ein Bit für jedes Pixel des Bildes und jedes Bit wird bei null initialisiert. Das Ordnen der Farbebenen kann vorher vorgenommen werden, so daß die erste Bitebene die dichteste Bitebene ist; dieser Prozeß kann jedoch auch durchgeführt werden, ohne daß die Bitebenen überhaupt geordnet werden, obwohl ohne ein Ordnen mehr Codieroperationen erforderlich sein können. Als Alternative hierzu können die Farben umgeordnet werden, so daß die Farbwerte in der Häufigkeitsreihenfolge vorliegen.

Die Farbebenen sind jedoch geordnet; das erste Pixel der Farbebene C wird gelesen (Schritt S2) und der Ebenen-Separator 212 prüft die Archiv-Anordnung, um zu sehen, ob das Pixel bereits decodiert worden ist (Schritt S3). Wenn die Archiv-Anordnung ein "1" Bit für dieses Pixel hat, ist es bereits codiert worden, andernfalls wird das Bit "0". Wenn das Pixel nicht bereits codiert worden ist, wird es codiert (S4) und das Archivbit für dieses Pixel wird "1" gesetzt (S5).

Nachdem das Pixel codiert ist oder nachdem herausgefunden ist, daß es bereits codiert ist, überprüft der Ebenen-Separator 212 mehr Pixel in der laufenden Farbebene (S6). Wenn [illegible]hr Pixel gibt, wird das nächste Pixel erhalten (S7), und der Prozeß wird von dem Schritt S3 an wiederholt. Im übrigen prüft der Separator 212, um zu sehen, ob eine Schwellenwertzahl von Pixel oder Farbebenen verarbeitet worden ist (S8). In diesem Fall werden die verbleibenden Pixel mit Hilfe einer Bitebenen-Trennung codiert (S9). Anderenfalls wird C inkrementiert, um die nächste Farbebene anzuzeigen (510). Der Ebenen-Separator 212 prüft dann, um zu sehen, ob es weniger als eine verbleibende Farbebene gibt [C>(MAX − 1)] (Schritt S11). Wenn dem nicht so ist, endet der Codierprozeß; andernfalls kehrt er auf den Schritt S2 mit dem ersten Pixel der nächsten Farbebene zurück. In einigen Ausführungsformen gibt es keinen zu erreichenden Schwellenwert, und das ganze Bit wird bei der Farbebenen-Trennung verarbeitet (d. h. Schritt S9 wird niemals erreicht). In anderen Ausführungsformen werden bei der Farbebenen-Trennung bzw. Aufteilung Pixel in Farbgruppen-Ebenen aufgeteilt, die mehr als eine Farbe pro Farbgruppe aufweisen. In diesen Ausführungsformen wird ein Schritt, restliches Codieren für eine Bitebene im Anschluß an den Schritt S11 vor Erreichen des Schrittes S2 (wahlweise Schritt 512) durchgeführt.

Da ein Bild mit einer großen Anzahl Farben (256 oder mehr) eine große Anzahl an Durchläufen für ein Codieren oder Decodieren erfordern kann, ist es zweckmäßig, wenn diese Operationen parallel in einer Parallel-Hardware durchgeführt werden können. Bei den vorstehend beschriebenen, hochschnellen binären Entropie-Codierern kann ein Entropie-Codieren parallel erfolgen, während nur eine einzige komprimierte Ausgangsdatei erzeugt wird, welche die notwendige Ausgangsgröße für die meisten Systeme ist. Für parallele Operationen, um einen Geschwindigkeitsvorteil zu schaffen, müssen die Kontext-Modelleinheiten und Codierer in der Lage sein, parallel zu arbeiten, d. h. Daten ohne Bezugnahme auf andere Kontext-Modelleinheiten oder Codierer zu verarbeiten oder ohne die Notwendigkeit hierauf zu warten. Natürlich kann ein Parallel-Codiersystem mit N parallelen Wegen nicht immer N-mal so schnell wie ein einziger Weg betrieben werden, da einige Wege auf andere Wege warten können, um benötigte Informationen zu erzeugen oder um einen Zugriff auf einen Speicher zu beenden, bei welchem ein Konkurrenzsituation vorliegt.

Wenn ein hochschneller binärer Entropie-Codierer bei einem Bitebenen-Codierer verwendet wird, ist ein paralleles Kontext-Modelliersystem, bei welchem jede Kontext-Modelleinheit auf einer Bitebene arbeitet, ein einfacher Weg für paralleles Verarbeiten, insbesondere wenn die Daten vorverarbeitet werden, so daß ein Aufspalten in die Bitebenen den Kompressionswirkungsgrad nicht nachteilig beeinflußt, wie in der am 23. Februar 1994 eingereichten US-Patentanmeldung S.N. 08/200,233 der Anmelderin der vorliegenden Anmeldung beschrieben ist, deren Erfinder Zandi et al sind und welche den Titel trägt "Compression of Palettized Images and Binarization for Bitwise Coding of M-ary Alphabets Therfor". (Auf diese Anmeldung wird nachstehend als "Zandi"-Patentanmeldung Bezug genommen). Jedoch müssen bei dem Farbebenen-Codieren die Daten durch ein anderes Kriterium als durch den Pixelwert getrennt werden.

Nachstehend werden mehrere Systeme für ein paralleles Codieren in Farbebenen mit Hilfe von Kontext-Modellen beschrieben. In einem dieser Systeme ist N die Anzahl von verfügbaren Parallel-Codierern. In einem

System wird das Bild in N Pixelb[illegible]n oder andere Unterteilungen des Bildes aufgeteilt, und jedes Band oder jede Unterteilung wird mittels eines Codierers codiert. Die Unterteilung in Bändern ist herkömmlich und begrenzt den Verlust im Kompressionsverhältnis, da jeder Codierer weniger als die durch das volle Bild geschaffene Information hat.

Wenn das Bild in Bänder unterteilt ist, dann werden Kontext-Pixel, welche sonst in einem anderen Band als dem zu codierenden Pixel liegen, durch bekannte Pixel ersetzt, wie beispielsweise Pixel, welche die häufigste oder erste Farbe haben, so daß ein Kontext für Pixel in einem Band nur durch Pixel in dem Band bestimmt wird.

Fig. 6 ist ein Blockdiagramm eines solchen Parallel-Codiersystems 600. Ein Bildpuffer 602, welcher die Anordnung an Pixel-Farbwerten enthält, welcher das Bild darstellen, ist logisch in N Bänder von jeweils 180 Zeilen unterteilt. Dieser Bildpuffer ist entweder ein Speicher, der gesondert von der Eingangsdatei 206 betrieben wird, oder eine Bilddatei 206 wird direkt betrieben, und ein Bildpuffer 206 ist lediglich eine andere logische Betrachtungsweise. Ein Band wird in eine Kontext-Modelleinheit 604 eingegeben, und die zwei Ausgangswerte der Kontext-Modelleinheit werden in einen Codierer 605 eingegeben. Die Ausgangswerte des Codierers 605 (0)-(N-1) werden in eine Bitstrom-Kombiniereinheit 606 eingegeben, welche einen einzigen komprimierten Datenstrom an die Datei 208 abgibt. Die Arbeitsweise der Bitstrom-Kombiniereinheit 606 ist ähnlich derjenigen, die in dem weiter oben angeführten US-Patent 5,381,145 dargestellt ist.

Eine Kontext-Modelleinheit 604 ist im einzelnen einschließlich einer Steuerlogik 608 und einem Puffer 610 zum Halten einer festen Kontextzeile dargestellt. Die Kontext-Modelleinheit 604 arbeitet wie eine herkömmliche Kontext-Modelleinheit, indem sie ein Kontextbit ID und ein Ergebnisbit an einem Codierer 605 vorsieht, außer wenn die Kontext-Modelleinheit 604 einen Kontext schafft, welcher sonst von Pixel außerhalb dieses Bandes abhängen kann. In diesen Fällen benutzt die Steuerlogik 608 Pixelwerte von dem Puffer 610, um den Kontext statt dessen durch Verwenden von Pixel von außerhalb des Bandes zu bilden. Diese Substitution macht das Codieren, folglich auch das Decodieren jedes Bandes unabhängig von den Pixel in den anderen Bändern.

Wenn Kontext durch Pixel links von und oberhalb des aktuellen Pixels vorgesehen ist, gibt die Kontext-Modelleinheit 604 diese Pixel entweder von einem internen Speicher oder von dem Bildpuffer 602 aus ein. Beispielsweise kann eine Kontext-Modelleinheit 604 (1) normalerweise auf Zeile 127 Bezug nehmen, um Kontext für Pixel in Zeile 128 festzulegen. Jedoch benutzt sie den Puffer 610 als einen Ersatz für die Zeile 127. Wenn die ersetzte Zeile eine Zeile von Pixel ist, welche alle dieselbe Farbe beispielsweise die häufigste Farbe haben, braucht der Puffer 210 nur groß genug zu sein, um einen Farbenwert zu speichern. In anderen Ausführungsformen ist der Inhalt des Puffers 610 im vorhinein festgelegt oder ist auf andere Weise ohne Bezugnahme auf andere Bänder bestimmbar.

Im Hinblick auf den Verlust an echter Kontext-Information infolge der Substitution wird der Kompressionswirkungsgrad etwas reduziert; jedoch ist er nur um eine von 128 Zeilen reduziert. Wenn die Bänder jeweils so dick wie 128 Zeilen sein können oder wenn Speicherraum für den Bildspeicher 602 nicht verfügbar oder unerschwinglich teuer ist, dann kann eine andere Einrichtung, um eine Parallelo[illegible]tion durchzuführen, besser sein.

In einigen Übertragungssystemen, insbesondere in Verbindung mit WWW-Bildern ist die Latenz von Teilen eines Bildes ein solches Ergebnis. Beispielsweise ist es oft wünschenswert, Pixel anzuzeigen, wenn sie empfangen werden, selbst wenn das gesamte Bild noch nicht empfangen ist. Bei dem in Fig. 6 dargestellten System wird jedes Band codiert und daher mehr oder weniger gleichzeitig decodiert. Um eine Latenz zu reduzieren und um folglich das Anzeigen der oberen Teile des Bildes zuerst zu ermöglichen, kann ein Parallelcodierer entsprechend konfiguriert werden, um das Bild in der Reihenfolge von oben nach unten zu verarbeiten.

Beispielsweise kann ein Codierer nur für die Farbebene 0 codieren, indem mehrere Pixel vor einem anderen Codierer stehen bleiben, welcher für die Farbebene 1 codiert. Wenn weniger als N Codierer vorgesehen sind, beendet der erste Codierer eine Zeile und codiert diese Zeile für eine Farbebene N, der nächste Codierer codiert für eine Farbebene N+1, usw. Um hierfür eine Geschwindigkeitszunahme zu schaffen, muß der Speicher, der verwendet ist, um das Bild zu erhalten, mehrfache gleichzeitige Leseoperationen zulassen oder muß in mehrere Bänke unterteilt werden, die jeweils unabhängig von verschiedenen Codierern zugänglich sind.

Fig. 7 und 8 zeigen ein derartiges alternatives Parallel-Codiersystem 700. Verglichen mit dem Parallel-Codiersystem 600 verwendet das Parallel-Codiersystem 700 genauere Kontext-Information auf Kosten einer möglicherweise langsameren Paralleloperation. In diesem System durchläuft jede Kontext-Modelleinheit das gesamte Bild und erzeugt Kontext für eine oder mehrere Farbebenen, während in dem System 600 jede Kontext-Modelleinheit Kontext von jedem Pixel in dessen Band für jede Farbebene schafft. Wenn es zu einem Zeitkonflikt kommt, (bei einem Versuch, bei mehr als einer Kontext-Modelleinheit, eine Speicherstelle zu lesen) werden ein oder mehrere Kontext-Modelleinheiten für einen Verarbeitungszyklus verzögert, was dann der Grund ist, warum das Gesamtsystem langsamer sein kann als eine reine Paralleloperation. Um das Streitproblem Speicher zu mildern, wird das Bild in Speicherbänke zergliedert, um den Speicherweg zu verbreitern. Solange jede Zeile vollständig codiert wird, bevor die nächste Zeile codiert ist, und Pixel von der zu codierenen Zeile dem Kontext eines zu codierenden Pixel nicht zugeteilt werden, kommt es zu keinen Kausalitätsproblemen.

Wie dargestellt, werden Pixelwerte aus einem eingegebenen Bild 206 durch eine Kontext-Modell-Steuereinheit 207 gelesen. Die Steuereinheit 207 zerlegt die Pixelwerte in M Speicherbänke 701 (1)-(M) und steuert die Auswahl-Zeilen von N(M-zu-1) Multiplexern 706(0)-(N-1), um einen breiteren Speicherweg zu einer Anordnung von Kontext-Modelleinheiten 708(0)-(N-1) zu schaffen. Jede Kontext-Modelleinheit 708 hat zwei Ausgänge, einen für ein Kontextfach ID und einen für ein Ergebnisbit. Diese Ausgänge führen zu einem parallelen Entropie-Codierer 710 (wie in Fig. 8 dargestellt ist).

Fig. 8 zeigt einen Kontext-Modelleinheit 708(i), einen Codierer 710 und eine Kontext-Modell-Steuereinheit 702 einschließlich eines einzigen Auswahl-Ausgangs und eines einzigen Pausenausgangs der Kontext-Modell-Steuereinheit 702 im einzelnen. Wie aus den Figuren zu ersehen ist, ist jeder Multiplexer 206 mit einem entsprechenden Auswahl-Ausgang einer Kontext-Mo-

dell-Steuereinheit 702 verbunden, jede Kontext-Modelleinheit 708 ist mit einem entsprechenden Pause-Ausgang einer Kontext-Modell-Steuereinheit 702 verbunden. Der Zweck des Auswahl-Ausgangs besteht darin, einen Multiplexer 706(i) zu steuern, so daß die Kontext-Modelleinheit 708(i) Eingangspixel erforderlichenfalls von der richtigen Speicherbank 704 erhält. Wie aus der vorstehenden Beschreibung zu ersehen ist, könnte das Parallel-Codiersystem ohne die Anzahl Speicherbänke 704(1)-(M) und die Multiplexer 706(1)-(M) betrieben werden. Der Zweck der Pausierausgänge besteht darin, daß die Kontext-Modelleinheit 708(i) pausiert, wenn diese Kontext-Modelleinheit im Begriff ist, ein Kontextmodell basierend auf Pixelwerten von Pixeln zu bestimmen, welche nicht schon bereits codiert worden sind (und im Falle eines Decodierers, wenn die Kontext-Modelleinheit Pixelinformation anfordert, welche noch nicht decodiert worden ist).

Eine parallele Codierinformation verläuft folgendermaßen. Eine Kontext-Modelleinheit-Steuereinheit 702 liest Pixelwerten aus einer Eingangsdatei 206 und speichert Zeilen des Bildes in Speicherbänken 704(1)-(M). In diesem Beispiel hält jede Speicherbank acht Pixel und die Bänke werden wieder verwendet, sobald ihr Inhalt nicht mehr länger benötigt wird. Natürlich sollten, wenn ein Kontext bei mehr als zwei Reihen von Pixel vorgesehen ist, die Speicherbänke mehr als zwei Reihen versorgen. Der Multiplexer 706(i) sieht die Pixel der Kontext-Modelleinheit 708(i) vor. Da die Kontext-Modelleinheit-Steuereinheit 702 das Fortbewegen der Kontext-Modelleinheit 708(i) verfolgt, merkt sie, welche Pixel für eine Kontext verwendet werden und welche Pixel bereits codiert worden sind. Wenn ein Pixel für einen Kontext benötigt wird, aber noch nicht codiert worden ist, läßt die Kontext-Modell-Steuereinheit 702 die Kontext-Modelleinheit 708(i) pausieren, bis dieses Pixel codiert worden ist.

In dem Beispiel der Fig. 7 startet die Kontext-Modelleinheit 708(0) Bearbeitungspixel 0. .3 einer Zeile, codiert nur Pixel in der Farbebene 0, während die Kontext-Modelleinheit 708(1) nichts tut (da ihre Kontexte davon abhängen, ob Pixel 0. .3 in der Farbebene 0 sind oder nicht). Wenn die Kontext-Modelleinheit 708(0) auf Pixel 4. .7 übergeht, startet die Kontext-Modelleinheit 708(1) bei Pixel 0. .3, die von einer Speicherbank 704(1) vorgesehen sind. Hierdurch wird ein Speicherkonflikt vermieden, da sich die Kontext-Modelleinheit 708(0) auf der Speicherbank 704(2) bewegt hat, wo Pixel 4. .7 gespeichert sind.

Wenn beide Kontext-Modelleinheiten 708(0) und 708(1) einen Satz von vier Pixel beendet haben, werden die Auswahl-Eingänge von Multiplexern 706(0) und 706(1) geändert, so daß die Speicherbank 704(3) mit der Kontext-Modelleinheit 704(0) und die Speicherbank 704(2) mit der Kontext-Modelleinheit 704(1) verbunden wird.

Folglich codiert die Kontext-Modelleinheit 708(0) Farbe 0 für Pixel 8. .11, während die Kontext-Modelleinheit 708(1) Farbe 1 für Pixel 4. .7 codiert. Da die Speicherbank 704(1) nunmehr frei ist, verarbeitet die Kontext-Modelleinheit 708(2) die Pixel in dieser Bank für die Farbebene 2. Wenn die Anzahl Farben größer ist als die Anzahl an Kontext-Modelleinheiten, dann verarbeitet die Kontext-Modelleinheit 708(0) Farbebenen 0, N, 2N, usw. während die Kontext-Modelleinheit 708(1) Farbebenen 1, N + 1, 2N + 1, usw. verarbeitet.

Für jedes Pixel, das nicht schon codiert worden ist, durchläuft eine Kontext-Modelleinheit eine Kontextinformation und ein Erg zeigt an, ob das aktuelle Pixel in der laufenden ebene ist oder nicht. Diese Information durchläuft einen Codierer, welcher die Information so, wie oben erläutert, codiert. Sobald eine Zeile verarbeitet ist, wird eine andere Zeile aus der Eingangsdatei in die Speicherbänke 704(1)-(M) gelesen. Wenn jede Zeile verarbeitet ist und die darin befindlichen Pixel codiert sind, werden die Pixel in der aktuellen Farbebene mit Flags versehen, um anzuzeigen, daß sie zu einer bereits verarbeiteten Farbebene gehören, so daß spätere Kontext-Modelleinheiten für spätere Farbebenen diese Pixel überspringen können. Wenn eine Kontext-Modelleinheit und ein Codierer eine Anzahl Pixel überspringen können, können sie bereit sein, den nächsten Satz von vier Pixel zu verarbeiten, bevor die Kontext-Modelleinheit und der Codierer vor ihnen die vorderste Bank Pixel verarbeitet haben. Wenn dies vorkommt, gibt die Kontext-Modell-Steuereinheit 702 ein Pausiersignal an die folgende Kontext-Modelleinheit ab. Der Codierer braucht dann nicht zu pausieren, da das Tempo durch das Ausgangssignal der Kontext-Modelleinheit bestimmt wird.

Bei dem gerade beschriebenen Beispiel werden vier Pixel pro Bank M Speicherbänke und N Codierer verwendet; jedoch kann selbstverständlich auch eine andere Anzahl bei diesen Komponenten verwendet werden. Mit Hilfe dieses Systems können eine Vielzahl von Kontext-Modelleinheiten parallel arbeiten, um einen Kontext für Pixel in einem Bild bei einer Farbebene zu schaffen, in dem eine Anzahl Speicherbänke verwendet werden und die Startzeitpunkte der Kontext-Modelleinheiten gestaffelt werden, so daß sie ein Verarbeiten einer Speicherbank annähernd zu demselben Zeitpunkt starten, an welchem sich die Kontext-Modelleinheit vor ihr zu der nächsten Speicherbank bewegt. Wenn eine Kontext-Modelleinheit in der Lage ist, Pixel schneller zu bearbeiten, wenn sie beispielsweise viele Pixel antrifft, welche bereits verarbeitet worden sind und wenn sie als nächstes ein Pixel aus einer Speicherbank unter Verwenden der Kontext-Modelleinheit vor ihr lesen will, wird diese Kontext-Modelleinheit pausieren, bis die Speicherbank frei ist.

In dem gerade beschriebenen Beispiel arbeiten die Codierer (oder genauer gesagt, die Kontext-Modelleinheiten und die Codierer) mit unabhängigen Geschwindigkeiten, da Codierer Pixel überspringen, welche in vorher codierten Farbebenen liegen. In noch einer weiteren Variation eines Parallel-Codiersystems werden die Codierer in entsprechender Weise miteinander verbunden. Obwohl dies die Parallelität etwas verringern kann, da einige Codierer auf andere warten, ist weniger Hardware erforderlich, da die Codierer nicht "hineinlaufen" ("run into"). Wenn die Codierer miteinander verbunden sind, benötigt das Parallel-Codiersystem nicht die Speicherbänke und Multiplexer. Wenn die Multiplexer getränkt werden, so daß N Parallel-Codierer immer verschiedene Pixel in einem Pixelfenster bearbeiten, stellt die Speicher-Bandbreite kein Problem dar.

Fig. 9 ist ein Blockdiagramm eines solchen vereinfachten Parallel-Codierers 900. In dem Parallel-Codierer 900 liest eine Steuereinheit 902 Pixelwerte aus der Ausgangsdatei 206 und durchläuft Pixelwerte bei jedem Taktzyklus in einer ersten Kontext-Modelleinheit 904(0), und die Pixelwerte durchlaufen ihrerseits andere Kontext-Modelle 904. Logischerweise tasten die Kontext-Modelleinheiten das Bild ab, wobei die eine Kontext-Modelleinheit 904(0) in Führung liegt und die anderen Kontext-Modelleinheiten ihr folgen. Ein Router 909

ist vorgesehen, um die Kontext-Modelleinheiten 904 mit Codierern 910(0)-(N-1) zu verbinden.

Im einfachsten Fall verbindet der Router 909 einfach jede Kontext-Modelleinheit geradewegs mit dem entsprechenden Codierer. Um jedoch Speicher in den Codierern zu sparen, kann die Reihenfolge der Codierer umgeordnet werden. Bei dem Speichersparen wird jede Farbe nur einem Codierer zugeordnet, da jeder Codierer einen Speicher für Kontextfächer enthalten muß, und ein Satz Kontext-Fächer für jede Pixelfarbe benötigt wird. Wenn folglich nur ein Codierer eine vorgegebene Farbe behandelt, müssen die Kontextfächer für diese Farbe nicht um mehr als einen verdoppelt werden. Wenn jeder Codierer auf bestimmte Farben beschränkt ist, können jedoch die Codierer an Pixel außerhalb der Reihenfolge arbeiten. Dies wird im einzelnen nachstehend in Verbindung mit Fig. 10 bis 12 erläutert. Wenn die Codierer 210 außerhalb der Reihenfolge sind, leitet der Router 909 die Kontextbits und das Ergebnisbit zu dem richtigen Codierer 910 weiter. Obwohl über den Router 901 verhindert werden könnte, daß bei den Kontext-Modelleinheiten 904 deren Reihenfolge umgeordnet würde, um den Codierern zu folgen, würde dadurch -Speicherplatz vergeudet, da jede Kontext-Modelleinheit mehr Kontextpixel verwalten müßte.

In dem dargestellten Beispiel erhält die Kontext-Modelleinheit 904(0) einen 9 Bit Kontext für ein aktuelles Pixel X von den neun, mit C0 bis C8 bezeichneten Pixels. Nur ein Kontextbit wird für jedes Pixel verwendet, nämlich ein Bit, das anzeigt, ob das Pixel in der aktuellen Farbebene liegt oder nicht. Von diesen neun Pixel sind vier Pixel in der Zeile über dem Pixel X, vier sind in der Zeile darunter und eines ist das Pixel links von dem Pixel X. Von dem Pixel über dem Pixel X sind zwei genau links davon und vier sind rechts davon. Die Abtastreihenfolge erfolgt natürlich von der oberen zu der unteren Zeile und von links nach rechts in einer Zeile. Alternativ hierzu können andere Bits verwendet werden, wie in Fig. 3 und 4 dargestellt ist.

Die Kontext-Modelleinheit 904(0) ist im einzelnen genauer dargestellt als die übrigen Kontext-Modelleinheiten 904(1)-(N-1). Jede Kontext-Modelleinheit 904 enthält einen Speicher für Kontextpixel, einen Speicher für die aktuelle Farbebene (CC), die von dieser Kontext-Modelleinheit zu verarbeiten ist, einen Vergleicher 908 und zwei Ausgänge, nämlich für einen Kontext und einen für ein Ergebnis. Der Kontext für Pixel X ist nicht der gesamt Pixel-Kontext, sondern gerade ein Bit für jedes Pixel, das anzeigt, ob es in der aktuellen Farbebene liegt oder nicht. Der Vergleicher 908 wird verwendet, um den Kontext aus dem CC-Wert und die Kontext-Pixel C0 bis C8 zu erzeugen. Der Vergleicher 908 wird auch verwendet, um das Ergebnisbit zu erzeugen, welches ein Bit ist, das anzeigt, ob das Pixel X die Farbe CC hat oder nicht. Obwohl nur ein Bit für jedes Kontextpixel verwendet wird, wird das ganze Pixel gespeichert, so daß es die Kontext-Modelleinheit 904(0) bis zu der Kontext-Modelleinheit 904(1) durchlaufen kann, welche das Pixel mit einem anderen CC-Wert vergleicht.

Wie in Fig. 9 dargestellt, passiert die Steuereinheit 902 das aktuelle Pixel X und zwei Kontext-Pixel (A, B) zu der Kontext-Modelleinheit 904(0), welche die Pixel als X, C4 bzw. C8 einsetzt. Nachdem die Kontext-Modelleinheit 904(0) das Codieren für das Pixel X beendet hat, nimmt es andere drei Pixel an, verschiebt jedes dieser Kontextpixel und dessen aktuelles Pixel um eins nach links, während sie die Pixel C0, C7 und C3 an die Kontext-Modelleinheit 904(1) abgibt, welche ihrerseits diese Pixel in ihre Position X, C8 bzw. C4 verschiebt. Eine optionale Verzögerung 609 wird in Fällen angewendet, in welchen ein Codierer eine gewisse Latenz, üblicherweise infolge von Pipelining aufweist. Wenn ein Decodierer Latenz hat, dann muß der Codierer eine entsprechende Latenz aufweisen, um auf diese Weise die Kausalität nicht zu verletzen. In diesen Fällen operieren die Kontext-Modelleinheiten 904(0) bis (N-1) nicht an aufeinanderfolgenden Pixeln, sondern an Pixeln, die einen gewissen Abstand voneinander haben. Die optionale Verzögerung 906 wird verwendet, um die Zwischenpixel zu speichern, bis sie von der nächsten Kontext-Modelleinheit benötigt werden. Wenn N, die Anzahl an Kontext-Modelleinheiten 904 kleiner ist als die Anzahl an Farben, kann der Parallel-Codierer 900 eine Einrichtung enthalten, um die Pixel, die aus der Kontext-Modelleinheit 904(N-1) herausgeschoben worden sind, in die Kontext-Modelleinheiten 904(0) zurückzuführen.

Ein Parallel-Codierer 900 könnte durch einen entsprechend programmierten digitalen Signalprozessor implementiert werden. In einer derartigen Ausführungsform ist die Arbeitsweise des Parallel-Codierers 900 ähnlich; es können jedoch verschiedene Techniken angewendet werden, um die Ausführung des Parallel-Codierers zu optimieren. Bei Hardware-Ausführungsformen des Parallel-Codierers 900 gibt es einen Geschwindigkeitsvorteil bei Pipelining- und Holdaten von den langsamsten Elementen aus bei größeren Bandbreiten. Bei den Software-Ausführungsformen wird der Geschwindigkeitsvorteil dadurch erhalten, daß Operationen identifiziert werden, die nicht getan werden müssen und daß diese Operationen nicht durchgeführt werden. Beispielsweise kann in der Hardware-Ausführungsform jeder Codierer gleichzeitig vier Pixel bedienen, um die Zeit zu minimieren, die ein Codierer arbeiten muß, um neue Pixelwerte wieder aufzufinden, wobei die Steuereinheit 902 verfolgt, welche Pixelwerte von welchem Codierer benötigt werden.

Fig. 10 bis 12 sind Beispiele von Schemata für ein paralleles Verarbeiten von Pixel in einem Bild. Jede Figur zeigt das Weiterbewegen von vier Codierern über den Satz Pixel oder ein Bild, das einen unterschiedlichen Satz Regeln anwendet, wie ein Codierer (genauer gesagt, eine Kontext-Modelleinheit/Codierer) über die Pixel fortschreitet. Jede dieser drei Figuren ist ein Gitter, das zeigt, welche Codierer an welchen Pixeln in welchen Taktzyklen arbeiten. Die oberste Zeile des Gitters zeigt die Pixel-Farbwerte für zehn Pixel, nämlich einen pro Spalte. Die Pixel-Farbwerte sind ausgewählte ganze Zahlen von 0 bis 15. Eine Gruppe von Zahlen der Form A:B an dem Schnittpunkt einer Pixelspalte und einer Taktperiodenzeile zeigt an, daß während dieses Taktzyklus Codierer A das Pixel überprüft, um zu sehen, ob es in der Farbebene B liegt. Die Schnittstellen, wo B zu dem Pixel-Wert paßt, ist mit einem "√" versehen, welcher den Zeitpunkt anzeigt, wenn der Wert des Pixels bestimmt worden ist und keine weiteren Operationen an diesem Pixel erforderlich sind. Ein Taktzyklus wird nicht benötigt, um die Farbe 15 zu codieren/ zu decodieren, da ein Pixel, das für die Farbe 14 zu prüfen ist, entweder Farbe 14 oder Farbe 15 ist, da alle übrigen Farben bereits geprüft worden sind und daher die Farbe nach dem Prüfen für Farbe 14 bestimmt werden muß.

Eine leere Schnittstelle zeigt an, daß das Pixel nicht von irgendeinem Codierer geprüft wird, und eine Schnittstelle, an welcher "leer" oder "warten" zu finden ist, zeigt an, daß ein Codierer Pixel zugeordnet worden

ist, aber er nichts mit ihm tut. So wird eine höhere Parallel-Codier-Durchführung erreicht, in dem die Anzahl an Taktzyklen erniedrigt wird, indem ein Codierer leerläuft oder wartet. In jeder dieser Figuren sind Codierer Farben zugeordnet, die in Tabelle 1 angezeigt sind. In Tabelle 1 ist die Farbe 15 keinem Codierer zugeordnet, da es die letzte Farbe ist, und da eine solche bei dem Farbebenen-Codieren nicht codiert wird.

Tabelle 1

Codierer	Farben
1	0, 4, 8, 12
2	1, 5, 9, 13
3	2, 6, 10, 14
4	3, 7, 11

Fig. 10 ergibt sich aus den Vorschriften, daß die vier Codierer an aufeinanderfolgenden Pixeln in einem Pixel-"Fenster" arbeiten, und die Codierer das Fenster nicht bewegen können, um ein neues Pixel abzudecken, bis das hintere Pixel in dem Fenster codiert/decodiert worden ist. Ein neues Pixel kann nur bei jedem Taktzyklus angenommen werden, da jedes neue Pixel mit einem Codieren für die Farbebene 0 beginnen muß, und der Codierer 0 der einzige Codierer für die Farbe 0 ist. Wie Fig. 10 zeigt, werden mit Hilfe dieser Vorschriften 10 Pixel in 24 Taktzyklen codiert, und das Überprüfen eines Pixels kann unterbrochen werden, da der Codierer 0 sich unmittelbar mit dem neuen Pixel bewegt, das in dem Fenster hinzugefügt worden ist. Jedes dieser Schemata genügt der Anforderung, daß für den Kontext eines Pixels in einer Farbebene, die in einer Decodierzeit verfügbar ist, in welcher das vorherige Pixel den Kontext beeinflußt, jedes Pixel vor dem aktuellen Pixel zumindest bis zu der aktuellen Farbebene decodiert sein muß. Dies kann bestätigt werden, indem bemerkt wird, daß die erste Aktion für jedes Pixel eine Operation von dem Codierer 0 ist, und daß jedes Pixel links von einem vorgegebenen Pixel bereits bis zu der Farbebene decodiert worden ist, die augenblicklich bei dem vorgegebenen Pixel angewendet wird.

Fig. 11 ergibt sich aus den Vorschriften, daß das Fenster bewegt wird, um ein neues Pixel zu bedecken, wenn das hintere Pixel codiert/decodiert wird, jedoch die Codierer nicht umgeordnet werden, so daß der Codierer 0 an dem vorderen Pixel arbeiten kann. Statt dessen wird mit dem Codieren des nächsten Pixels "gewartet", bis Codierer 0 verfügbar ist. Sobald das Prüfen an einem Pixel beginnt, wird es auf diese Weise betrieben, bis es codiert ist. Mit Hilfe dieser Vorschriften werden die 10 Pixel in nur 21 Taktzyklen codiert.

Fig. 12 ergibt sich aus den Vorschriften, daß die Codierer nicht an aufeinanderfolgenden Pixeln arbeiten müssen, sondern daß nur der Codierer 0 an einem neuen Pixel starten kann. Mit Hilfe dieser Vorschriften werden dieselben 10 Pixel in 20 Taktzyklen codiert (und andere Pixel werden teilweise codiert werden). Da das Fenster breiter sein kann, erfordern diese Vorschriften mehr Speicherplatz, um fortlaufend Pixel zu speichern. Die Größe dem Fensters ist durch die Tatsache begrenzt, daß der Codierer 0 auch an den hinteren Pixels für Farben außer der Farbe 0 arbeitet.

In der vorstehenden Beschreibung wurde das zu codierende Bild in einzelne Bildebenen aufgeteilt. Obwohl die Vorteile des Aufteilens in Farbebenen aus der vorstehenden Beschreibung ersichtlich sind, kann es Situationen geben, bei welchen die zusätzliche Komplexibilität, viele Durchläufe über dem Bild durchzuführen, vermieden werden muß; in diesen Fällen kann ein Hybrid zwischen der Aufteilung in Bitebenen und der Aufteilung in Farbebenen verwendet werden.

In einem solchen Hybrid wird das Bild nicht in eine Farbebene pro Farbe aufgeteilt, sondern es wird in eine Farbgruppen-Ebene für jede Gruppe von zwei Farben aufgeteilt, und die Farbgruppen-Ebene wird dann als eine Bitebene codiert. In einem anderen Hybrid enthält jede Farbgruppe vier Farben und die Farbgruppen-Ebene wird in der zwei Bitebenen aufgeteilt. In den in Fig. 1 dargestellten Beispiel kann jedes Pixel eine von sechzehn Farben haben. Bei einer geraden Farbaufteilung wird das Bild in sechzehn Farbebenen aufgeteilt, und ein Codierer macht bis zu fünfzehn Durchläufe über dem Bild (wobei die letzte Farbe als letzte codiert werden muß). Mit einer hybriden Farbaufteilung kann das Bild von Fig. 1 in acht Gruppen von zwei Farben oder in vier Gruppen von jeweils vier Farben aufgeteilt werden. Bei acht Farbgruppen macht der Codierer meistens sieben Durchläufe über dem Bild, um jede Pixel-Farbgruppe zu bestimmen, und dann noch einen Durchlauf mehr, um die Farbe jedes Pixel in der Pixel-Farbgruppe zu bestimmen, und macht somit insgesamt acht Durchläufe. Bei vier Farben pro Gruppe macht der Codierer bis zu drei Durchläufen, um die Farbgruppe zu bestimmen, macht dann zwei Durchläufe (bei der Bitebene), um die Farbe in der Gruppe zu bestimmen, macht also insgesamt noch vier Durchläufe. In einem anderen Extremfall, bei welchem keine Farbteilung vorgenommen wird, erfordert das Bitebenen-Codieren vier Durchläufe über dem Bild. Obwohl der Unterschied zwischen dem Bitebenen-Codieren und der Farbebenen-Aufteilung nur fünfzehn Durchläufe gegenüber vier Durchläufen ist, wird bei mehr Farben der Unterschied größer. Beispielsweise erfordert bei einer 8 Bit Farbe (256 Worte) eine volle Farbebenen-Aufteilung bis zu 255 Durchläufen; zwei Farben pro Gruppenaufteilung fordern nicht mehr als 128 Durchläufe (127 + 1); vier Farben pro Gruppenaufteilung erfordern nicht mehr als 65 Durchläufe (63 + 2), während eine Bitebenen-Aufteilung 8 Durchläufe erfordert. Folglich ist bei Bildern mit einer größeren Anzahl Farben ein Parallel-Codieren öfters erforderlich.

Bei noch einer weiteren Abhandlung werden Pixel-Farbwerte durch Vektoren dargestellt, und jede Vektorkomponente wird unabhängig bezüglich der Farbe aufgeteilt. Beispielsweise können bei 8 Bit Pixel-Farbwerten die Pixelwerte auf zwei 4 Bit-Werte verteilt werden, um einen zweidimensionalen Vektor für jedes Pixel zu bilden. Jeder dieser 4 Bit Unterfarben-Komponenten wird in eine von sechzehn Farbebenen aufgeteilt, die der entsprechenden Dimension des Vektors zugeordnet sind. Diese Variation ist für Parallel-Decodieren anwendbar. Dies erfordert ein Maximum von 30 Durchgängen oder 15 Durchgängen pro Komponente.

Bezugszeichenliste

Zu Fig. 1
7, 12, 3 Farbebene

Zu Fig. 2
206 Bilddatendatei
212 Ebenen-Separator
214 Kontext-Modelleinheit

215 Kontext
Ergebnis
216 Entropie-Codierer
217 Komprimierter Bitstrom
210 Dekomprimierter Bilddatendatei
222 Ebenen-Akkumulator
220 Kontext-Modelleinheit
218 Entropie-Decodierer
208 Komprimierte Bilddatei

Zu Fig. 5
S1 C=0 Initiieren; Historie initiieren;
S2 Erster Pixel von Ebene 10 erhalten
S3 Ist Pixel bereits codiert worden?
S4 Aktuelles Pixel codiert?
S5 Historie aktualisiert?
S6 Mehr Pixel in Ebene C?
S7 Nächstes Pixel erhalten
S8 Schwellenwert erreicht?
S9 Restliche Pixel durch Bitebene codieren
S10 C Inkrementieren
512 Bitebene (N) Codieren, wenn Ebene C mehr als eine Farbe enthält

Zu Fig. 6
604(1), (2), (3), (N) Kontext-Modelliereinheit 0, 1, 2 ... N-1
605(1), (2), (3), ... (N) Codierer 0, 1, 2 ... N-1
606 Bitstrom-Kombiniereinheit
208 Komprimierte Bilddatei
610 Feste Kontextzeile
608 Steuerlogik
612 Kontextfach ID
Ergebnis

Zu Fig. 7
206 Eingegebenes Bild
702 Kontext-Modell-Steuereinheit
Auswählen Pause
708(1) ... (N) Kontext-Modelleinheit 1 ... N

Zu Fig. 8
701 Zwei Zeilen-Pixelpuffer
702 Kontext-Modell-Steuereinheit auswählen (i)
708(i) Kontext-Modelleinheit (i)
709 Kontextfach ID
Ergebnis
710(i) Codierer (i)

Zu Fig. 9
206 Eingabedatei
902 Steuereinheit
904(0) Kontext-Modelleinheit 0
906 Optionale Verzögerung
904(1), (2), ... (N-1) Kontext-Modelleinheit 1, 2, ... N-1
910(0), (1), (2) ... (N-1) Codierer 0, 1, 2, ... N-1.

Patentansprüche

1. Verfahren zum Komprimieren einer Anordnung von Pixelwerten, bei welchem die Anordnung kollektiv ein Bild erzeugt, und jedes Pixel durch einen Farbwert und eine Stelle in dem Bild gekennzeichnet ist, wobei das Verfahren die Schritte aufweist:
Pixel des Bildes in Farbebenen aufteilen, wobei eine Farbebene eine zweidimensionale Anordnung von binären Werten ist, die anzeigen, ob ein Farbwert für ein Pixel ein Farbwert ist, der einem Farbwert für diese Ebene en[illegible]ht;
eine erste Farbebene codieren;
Pixel einer zweiten Farbebene codieren, welche nicht in der ersten Farbebene gefunden werden, und
für eine spätere Farbebene Pixel wiederholt codieren, welche nicht eine Farbe haben, die einer vorher codierten Farbebene entspricht.

2. Verfahren nach Anspruch 1, bei welchem die Farbebenen nach der Farbhäufigkeit geordnet werden, wobei eine Farbebene mit einer Farbe, welche die häufigste Farbe in dem Bild ist, eine erste Farbe ist, und eine Farbebene, die der ersten Farbe entspricht, die erste Farbebene ist, wodurch die Farbebenen in der Häufigkeitsreihenfolge codiert werden.

3. Verfahren nach Anspruch 1, bei welchem eine Farbebene, welche Farben entspricht, die nicht in dem Bild vorhanden sind, und eine Farbebene, die einer Farbe entspricht, welche in dem Bild vorhanden ist, nicht codiert werden.

4. Verfahren nach Anspruch 1, bei welchem die Codierschritte Entropie-Codier-Schritte sind.

5. Verfahren nach Anspruch 4, bei welchem die Entropie-Codier-Schritte Codierschritte sind, bei welchen ein hochschneller binärer Entropie-Codierer verwendet wird.

6. Verfahren nach Anspruch 4, bei welchem die Entropie-Codierschritte die Codier-Schritte mit Hilfe eines Q-Codierers sind.

7. Verfahren nach Anspruch 4, bei welchem die Entropie-Codierschritte Codier-Schritte mit Hilfe eines B-Codierers sind.

8. Verfahren nach Anspruch 4, bei welchem das Entropie-Codieren durchgeführt wird, wobei ein Kontext zumindest einen Wert eines Pixels nahe dem zu codierenden Pixel enthält.

9. Verfahren nach Anspruch 4, bei welchem ein Entropie-Codieren durchgeführt wird, wobei ein Kontext zumindest ein Wert einer Position des zu codierenden Pixels enthält.

10. Verfahren nach Anspruch 4, bei welchem ein Entropie-Codieren zumindest einen Schritt aufweist, einen Kontext für ein aktuelles Pixel zu bestimmen, und der Kontext ein Hinweis darauf ist, welche Nachbarpixel in der zu codierenden Farbebene sind.

11. Verfahren zum Komprimieren einer Anordnung von Pixelwerten, bei welchem die Anordnung kollektiv ein Bild erzeugt und jedes Pixel durch einen Farbwert und eine Stelle in dem Bild gekennzeichnet ist, wobei das Verfahren die Schritte aufweist:
Ränder von Unterbildern identifizieren;
einen Kontext für jedes Pixel erzeugen, welcher Kontext auf einer Beziehung zwischen den Rändern und jeder Pixel-Stelle basiert, und
einen Entropie-Codierer verwenden, um jedes Pixel mit dem Kontext für dieses Pixel zu codieren.

12. Verfahren nach Anspruch 11, bei welchem der Kontext ein binärer Wert ist, welcher anzeigt, ob ein Pixel in einer ersten Zeile liegt oder nicht.

13. Verfahren nach Anspruch 12, bei welchem das Bild eine Anzahl von Schemen und die Ränder aufweist, die jedes der Anzahl Schemen begrenzen.

14. Parallel-Kontext-Modelleinheit, welche Kontextinformation für Pixel in einem Bild entweder in einem Entropie-Codierer oder in einem Entropie-

Decodierer schafft, aufweisend:
einen Bildpuffer, welcher das zu verarbeitende Bild hält, wobei das Bild eine Pixelwert-Anordnung in einem Speicher ist, wobei der Bildpuffer in N Unterbereiche unterteilt ist, und N größer als eins ist, und
N Kontext-Modelleinheiten, die jeweils einem der N-Unterbereiche zugeordnet sind und die jeweils eine Einrichtung aufweisen, um Ersatz-Kontextpixel darzustellen, wobei die Ersatz-Kontext-Pixel Kontext-Pixel ersetzen, welche außerhalb eines Unterbereichs liegen, aber Kontextpixel für Pixel in dem vorgegebenen Unterbereich sind.

15. Parallel-Kontext-Modelleinheit nach Anspruch 14, bei welcher jede Farbebene von einer einzigen der Anzahl Kontext-Modelleinheiten verarbeitet wird.

16. Parallel-Kontext-Modelleinheit, welche Kontextinformation für Pixel in einem Bild entweder in einem Entropie-Codierer oder in einem Entropie-Decodierer schafft, aufweisend:
eine Anzahl Speicherbänke, wobei jede Speicherbank eine Speicherung für eine Anzahl Pixel und Kontext-Pixel für die Anzahl Pixel aufweist;
eine Anzahl Kontext-Modelleinheiten, von denen jede mit einer der Anzahl Speicherbänke verbindbar ist, und jede Speicherbank mit einer der Anzahl Kontext-Modelleinheiten verbindbar ist, wobei jede der Anzahl Kontext-Modelleinheiten einen Pausiereingang für ein Pausieren der Kontext-Modelleinheit aufweist, und eine Kontext-Modelleinheit-Steuereinheit, welche aufweist:
einen Bildeingang für Pixel eines Bildes;
einen Speicherbus, um die Pixel in Speicherbänke der Anzahl Speicherbänke zu laden;
eine Anzahl Leitsignalleitungen zum Steuern einer spezifischen Verbindung von Speicherbänken mit Kontext-Modelleinheiten, und Ausgänge, damit Kontext-Modelleinheiten pausieren.

17. Parallel-Kontext-Modelleinheit nach Anspruch 16, bei welcher die Anzahl von Kontext-Modelleinheiten bis auf eine der Anzahl Speicherbänke durch eine Anzahl Multiplexer verbunden sind, welche von der Kontext-Modelleinheit-Steuereinheit gesteuert worden sind.

18. Parallel-Entropie-Codierer, aufweisend
einen Bildspeicher zum Speichern von Pixelwerten, die kollektiv ein zu komprimierendes Bild darstellen;
eine Codierer-Steuereinheit, die mit dem Bildspeicher verbunden ist, um aus ihm Pixelwerte zu lesen;
eine Anzahl Kontext-Modelleinheiten, die jeweils mit der Codierer-Steuereinheit verbunden sind, um Pixelwerte für Kontext-Pixel und ein aktuelles Pixel zu erhalten, wobei jede Kontext-Modelleinheit ein Ausgangssignal eines Ergebnisses und einen Kontext für das Ergebnis schafft, wobei das Ergebnis das Resultat eines Tests des aktuellen Pixels gegenüber einem Farbebenen-Wert ist, welcher der Kontext-Modelleinheit, die das Ergebnis abgibt, zugeordnet ist, und der Kontext die Kontextpixel mit dem Farbebenen-Wert vergleicht, wobei die Codierer-Steuereinheit das Fortbewegen jedes der Anzahl Kontext-Modelleinheiten über dem Bild steuert, um sicherzustellen, daß keine Kontext-Modelleinheit für einen Kontext ein Pixel benutzt, was nicht bereits zumindest teilweise codiert worden ist.

19. Verfahren zum Komprimieren einer Anzahl Pixelwerte aus einem komprimierten Datensatz, welches Verfahren die Schritte aufweist:
aus einer Apriori-Information und vorher dekomprimierten Pixelwerten eine aktuelle Farbebene für ein aktuelles, zu decodierendes Pixel zu identifizieren;
aus vorher decodierten Pixel in vorher decodierten Farbebenen eine Position in einer Anordnung des aktuellen, zu decodierenden Pixels zu bestimmen, wobei angenommen wird, daß die Position des aktuellen Pixels eine Position, nicht eine Position der vorher decodierten Pixel in den vorher decodierten Farbebenen ist, und
nachfolgende Pixel in nachfolgenden Farbebenen wiederholt decodieren.

20. Verfahren nach Anspruch 19, bei welchem der Schritt, Decodieren eines Pixels, gemäß einem adaptiven Entropie-Code vorgenommen wird, wobei der adaptive Entropie-Code durch Bezugnahme auf einen Kontext für die festgelegte Stelle für das aktuelle, zu decodierende Pixel bestimmt wird.

21. Verfahren nach Anspruch 19, bei welchem eine endgültige Farbebene nicht decodiert wird, und Pixelstellen, welche nicht einen Pixelwert enthalten, einem Pixelwert für die letzte Farbebene zugeordnet werden.

22. Parallel-Entropie-Decodierer, mit einem Bildspeicher zum Speichern decodierter Pixelwerte an Pixelstellen, wobei die Pixelwerte an den Pixelstellen kollektiv ein Bild darstellen;
einer Decodierer-Steuereinheit, die mit dem Bildspeicher verbunden ist, um aus diesem Pixelwerte zu lesen und um in diesen Pixelwerte zu schreiben;
einer Anzahl Kontext-Modelleinheiten, die jeweils mit der Codierer-Steuereinheit verbunden sind, um Kontext-Pixelwerte zu erhalten, wobei jede Kontext-Modelleinheit einen Kontext-Ausganswert schafft, welcher Kontext-Ausgangswert den Kontext eines aktuellen, zu decodierenden Pixel anzeigt, und der Kontext ein Satz von Vergleichswerten der Kontextpixel mit einem aktuellen Farbebenenwert ist;
eine Anzahl Codierer, die jeweils mit einer Kontext-Modelleinheit der Anzahl Kontext-Modelleinheiten verbunden sind, wobei jeder Codierer einen Kontext für eine aktuelle Farbebene verwendet, um ein Bit zu decodieren, das anzeigt, ob das aktuelle, zu decodierende Bit in der aktuellen Farbebene liegt oder nicht und
einer Einrichtung, damit die Bits mit der Codierer-Steuereinheit kommunizieren.

23. Verfahren zum Komprimieren einer Anordnung von Elementwerten, wobei die Anordnung von Elementwerten kollektiv einen Datensatz bildet, und jedes Element durch einen Elementwert und eine Stelle in der Anordnung gekennzeichnet ist, wobei das Verfahren die Schritte aufweist:
die Elemente in dünne Anordnungen aufteilen, wobei eine dünne Anordnung die Elemente enthält, die Elementwerte haben, die gleich einem Anordnungswert sind, welcher der dünnen Anordnung zugeordnet ist, wobei eine dünne Anordnung eine Anordnung von Binärwerten mit einer Eins-zu-Eins-Entsprechung zu der Anordnung von Elementen ist;
eine erste dünne Anordnung codieren;
einen Teil einer zweiten Farbebene codieren; wo-

bei der codierte Teil die Bits [redacted]lementenstellen von noch nicht codierten Elementen sind, und für nachfolgende dünne Anordnungen Elemente an Elementenstellen wiederholt codieren, welche nicht Elementenstellen für Elemente in vorher codierten dünnen Anordnungen sind.

24. Verfahren zum Dekomprimieren einer Elementanordnung aus einer komprimierten Datendatei, wobei die Elementanordnung kollektiv einen Datensatz mit Elementen bildet, die jeweils durch einen Elementwert und eine Stelle in der Elementanordnung gekennzeichnet sind, wobei das Verfahren die Schritte aufweist:

einen ersten Bereich der komprimierten Datendatei zu identifizieren, wobei der erste Bereich einen verdichteten Informationsinhalt einer ersten dünnen Anordnung darstellt, und die erste dünne Anordnung eine Anordnung von binären Werten ist, die anzeigt, welche Elemente der Elementanordnung Elementwerte haben, die gleich einem ersten dünnen Anordnungs-Elementwert sind;

die erste dünne Anordnung von dem ersten Bereich aus decodieren;

einen zweiten Bereich der komprimierten Datendatei identifizieren, wobei der zweite Bereich einen komprimierten Informationsinhalt einer zweiten dünnen Anordnung von Binärwerten darstellt, die jeweils anzeigen, ob ein Element der Elementanordnung einen Elementwert hat, welcher gleich einem zweiten dünnen Anordnungs-Elementwert ist, oder einen Elementwert hat, welcher sich von dem ersten dünnen Anordnungs-Elementwert mit dem zweiten dünnen Anordnungs-Elementwert unterscheidet, und wobei der zweite Bereich weniger als die gesamte Elementwert-Information für Elemente enthält, die in dem ersten dünnen Bereich gefunden worden sind;

die zweite dünne Anordnung aus dem zweiten Bereich dekodieren, und

die Schritte, Identifizieren eines Bereichs, Decodieren eines dünnen Bereichs, wiederholen, bis eine Schwellenwertzahl von dünnen Bereichen decodiert wird, wobei jede nachfolgende dünne Anordnung jedes noch nicht decodierte Element identifiziert, ob das Element einen Elementwert hat, welcher gleich dem dünnen Anordnungs-Elementwert ist.

25. Verfahren nach Anspruch 24, bei welchem die Schritte, Decodieren, Anwenden von Bitebenen-Decodieren, die Elementwerte für dünne Anordnungen aufweisen, welche noch nicht decodiert worden sind, wenn die Schwellenwertzahl von dünnen Anordnungen decodiert wird.

26. Verfahren nach Anspruch 24, bei welchen die Schwellenwertanzahl von dünnen Anordnungen eins niedriger ist als die Anzahl verschiedener Elementwerte, wobei das Verfahren ferner den Schritt aufweist, einen fehlerhaften Elementwert Element zuzuordnen, die in der Schwellenwertanzahl von Decodierungen von dünnen Anordnungen nicht decodiert worden sind.

27. Verfahren nach Anspruch 24, bei welchem jedes Element durch den Elementwert gekennzeichnet ist, der bei den Decodierschritten decodiert worden ist, und das Verfahren ferner die Schritte aufweist, Unterwerte für jedes Element mit Hilfe eines Bitebenen-Decodierens zu decodieren, nachdem die Schwellenwertzahl von dünnen Anordnungen decodiert ist.

28. Bildkompressor zum Komprimieren eines Bildes, um einen komprimierten Datensatz zu erzeugen, mit

einem Bildspeicher zum Speichern von Bildwerten, die kollektiv das zu komprimierende Bild darstellen, wobei jedes Pixel des Bildes durch einen Pixelwert und eine Pixelstelle in dem Bild gekennzeichnet ist;

einem Pixelwert-Leser, der mit dem Bildspeicher verbunden ist und welcher Pixelwerte aus dem Bildspeicher liest und binäre Darstellungen von Pixels in der Farbebenen-Reihenfolge abgibt, wobei eine binäre Darstellung eines Pixels in einer aktuellen Farbebene anzeigt, ob sich das Pixel in der aktuellen Farbebene befindet oder nicht, wobei ein Pixel in einer Farbebene ist, wenn und nur wenn sein Pixelwert in einem Satz von Pixelwerten ist, die der Farbebene zugeordnet sind;

einem Farbebenen-Codierer, der mit dem Pixelwert-Leser verbunden ist, um noch nicht codierte Pixel in der aktuellen Farbebene zu codieren, wobei der Farbebenen-Codierer aufweist:

eine Kontext-Modelleinheit, welche aus vorher codierten Pixel, die zumindest aus der aktuellen Farbebene oder einer der vorher codierten Farbebenen abgezogen sind, einen Kontext für ein aktuelles Pixel in einer aktuellen Farbebene identifiziert, wobei das Kontextmodell einen ersten Ausgangswert, welcher ein Ergebnis eines Vergleichs des Pixelwerts des aktuellen Pixel und der Farbe der aktuellen Farbebene anzeigt, und einen zweiten Ausgangswert aufweist, welcher den Kontext für das aktuelle Pixel anzeigt, und

einen binären Entropie-Codierer, der mit der Kontext-Modelleinheit verbunden ist und der ein aktuelles Pixel in einem Bitstrom codiert, welcher von dem binären Entropie-Codierer auf der Basis des Ergebnisses und des Kontexts abgegeben worden ist, der von der Kontext-Modelleinheit abgegeben worden ist, und

eine Bitstrom-Sammeleinrichtung, die mit dem binären Entropie-Codierer verbunden ist, um den Ausgangs des binären Entropie-Codierers in dem komprimierten Datensatz zu sammeln.

29. Bildkompressor nach Anspruch 28, bei welchem jeder Satz Pixelwerte nur einen Pixelwert enthält, wodurch alles, was einen gemeinsamen Pixelwert hat, auf die jeweilige Farbebene hinausläuft, die Pixel enthält.

30. Bildkompressor nach Anspruch 28, bei welchem zumindest ein Satz Pixelwerte eine Anzahl Pixelwerte aufweist, der Bildkompressor ferner den Bitebenen-Codierer zum Codieren von Information aufweist, die anzeigt, welche der Anzahl Pixelwerte der Pixelwert für Pixel in Farbebenen ist, die Sätze von mehreren Pixelwerten haben.

31. Bildkompressor nach Anspruch 28, bei welchem die Kontext-Modelleinheit einen Kontext schafft, der darauf basiert, ob benachbarte Pixel um das aktuelle Pixel herum in der aktuellen Farbebene sind oder nicht.

32. Bilddekompressor, um ein Bild aus einem komprimierten Datensatz zu dekomprimieren, um ein nicht-komprimiertes Bild zu erzeugen, wobei jedes Pixel des nicht-komprimierten Bildes durch einen Pixelwert und eine Pixelstelle in dem nicht-verdichteten Bild charakterisiert ist, wobei der Bild-De-

kompressor aufweist:
einen Speicher zum Speichern eines komprimierten Datensatzes; einen Farbebenen-Decodierer, um Pixel aus dem verdichteten Datensatz in der Farbebenen-Reihenfolge zu decodieren, wobei der Farbebenen-Decodierer aufweist:
eine Kontext-Modelleinheit, welche aus vorher decodierten Pixel, die zumindest aus einer aktuellen Farbebene oder zumindest aus einer der vorher decodierten Farbebenen abgezogen worden sind, einen Kontext für ein aktuelles Pixel in einer aktuellen, zu decodierenden Farbebene identifiziert, wobei der Kontext die Farbebenen für Pixel anzeigt, die dem aktuellen Pixel benachbart sind;
einen binären Entropie-Decodierer, der mit der Kontext-Modelleinheit und dem komprimierten Datensatz verbunden ist, welcher das aktuelle Pixel basierend auf dem Kontext für das aktuelle Pixel decodiert, das durch die Kontext-Modelleinheit identifiziert worden ist, und
einen Farbebenen-Akkumulator, welcher Pixel, die durch den binären Entropie-Decodierer bezüglich der Farbebene decodiert worden sind, in einem Bildspeicher speichert, wodurch das nicht-komprimierte Bild in dem Bildspeicher erzeugt wird.

33. Bild-Dekompressor nach Anspruch 32, bei welchem der Kontext für das aktuelle Pixel ein Kontext ist, der anzeigt, ob die Pixel, die dem aktuellen Pixel benachbart sind, in der aktuellen Farbebene sind oder nicht.

Hierzu 11 Seite(n) Zeichnungen

– Leerseite –

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

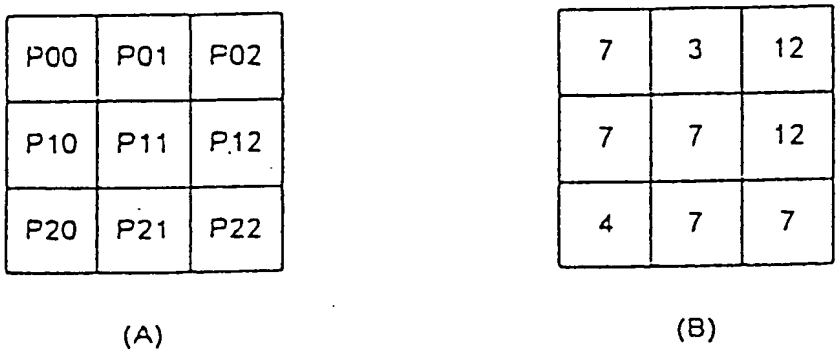


Fig. 1

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

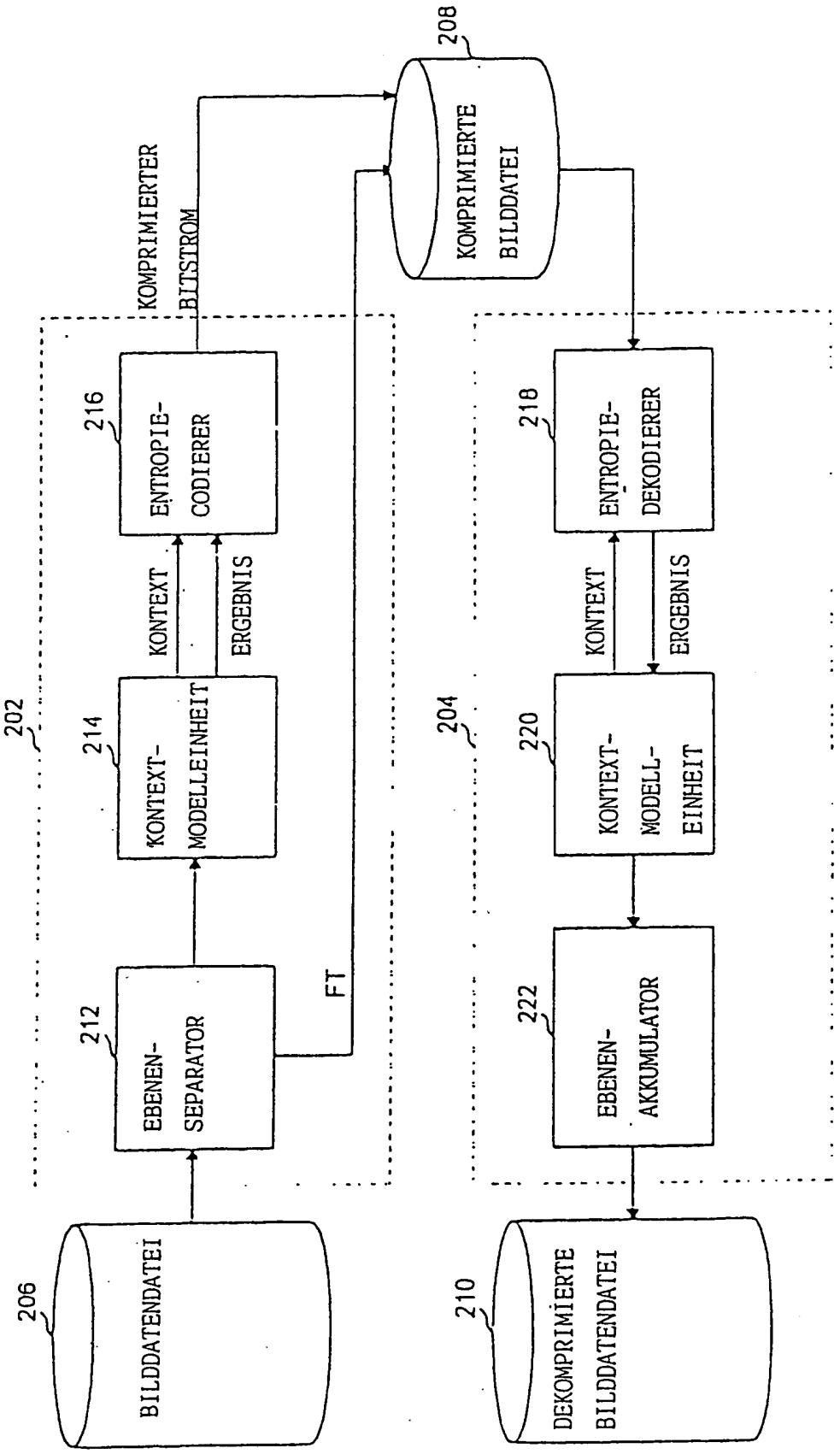


Fig. 2

602 039/561

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

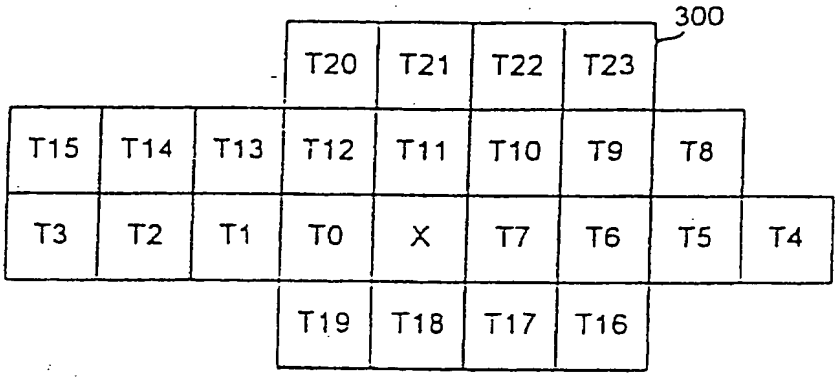


Fig. 3

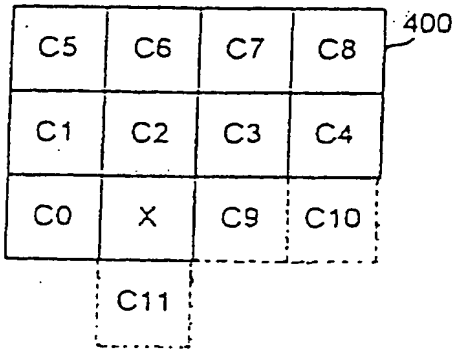


Fig. 4

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

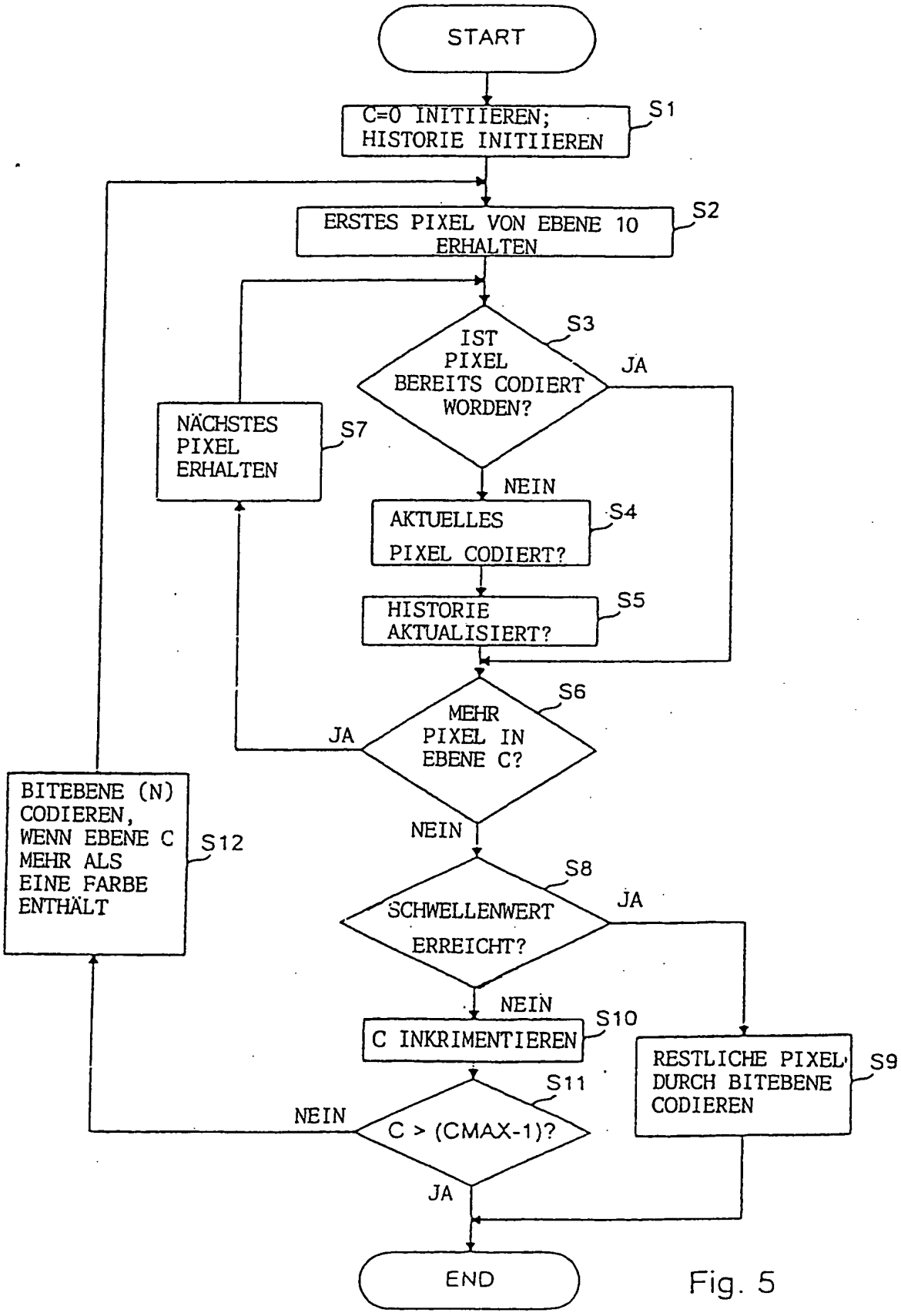


Fig. 5

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

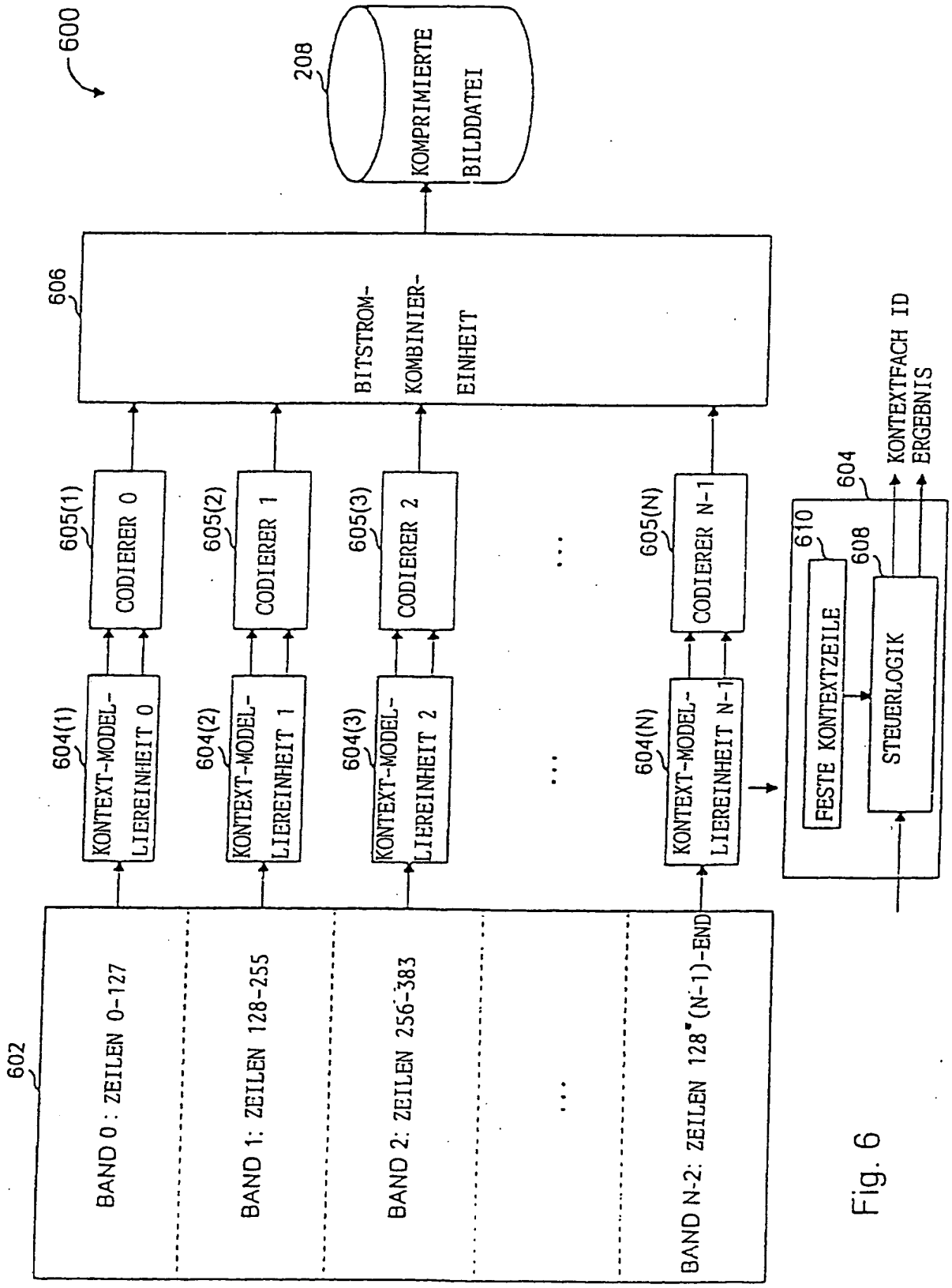


Fig. 6

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

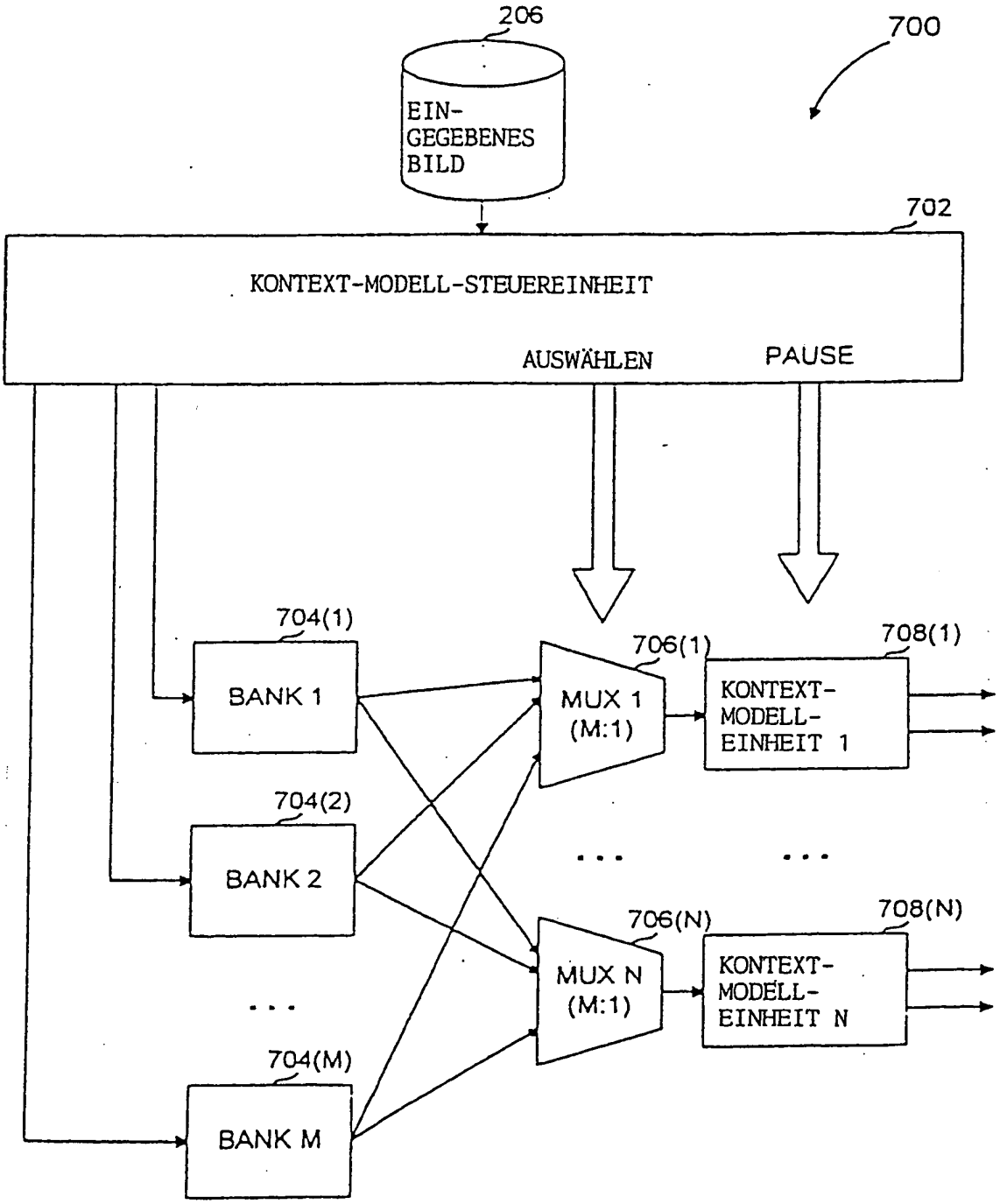


Fig. 7

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

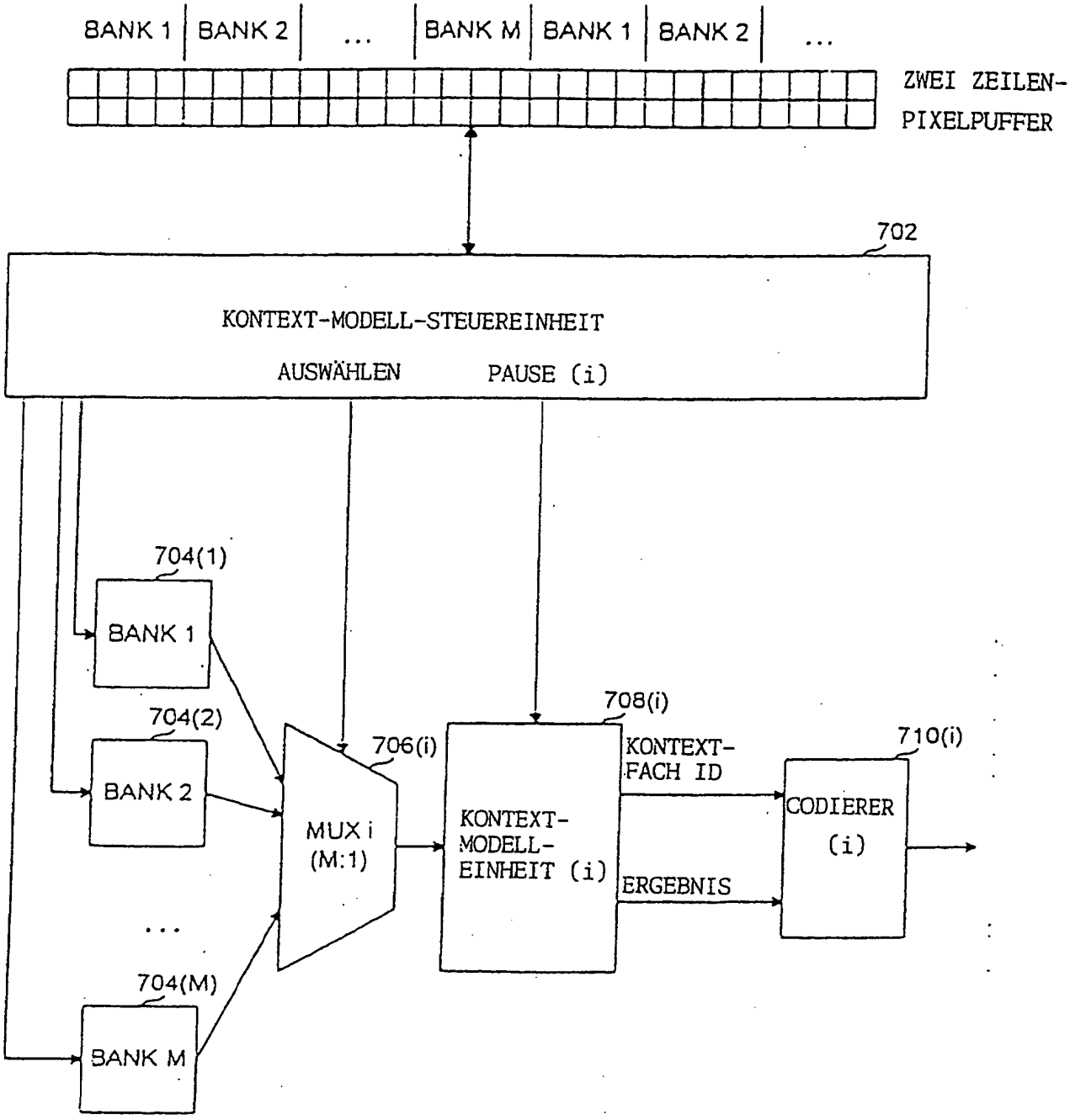
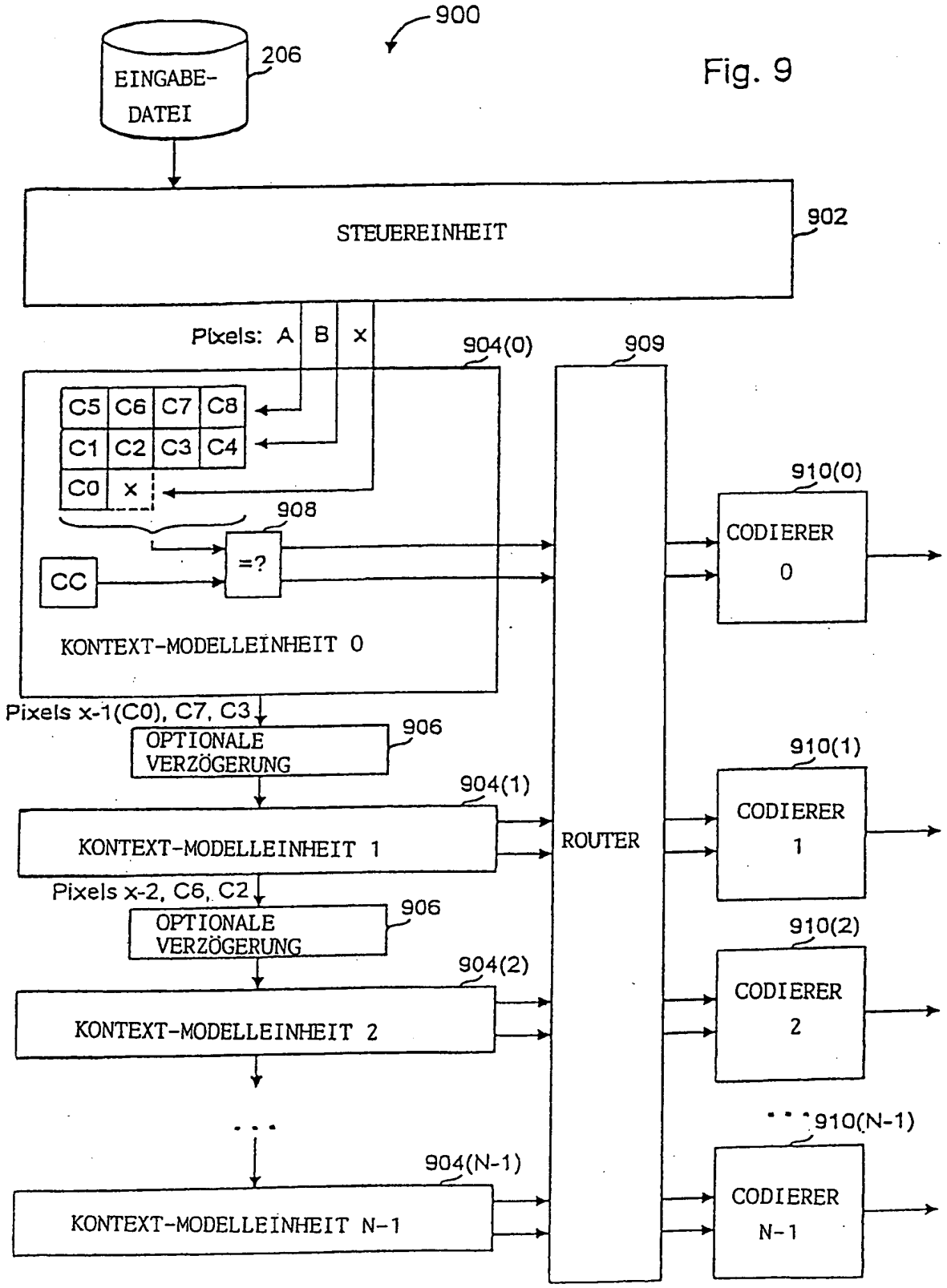


Fig. 8

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegung 26. September 1996

Fig. 9



Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

ZEIT	PIXELWERT									
	2	0	5	1	8	15	7	2	1	0
0	0:0									
1	1:1	0:0 ✓								
2	2:2 ✓	1:LEER	0:0							
3	3:LEER	2:LEER	1:1	0:0						
4		3:LEER	2:2	1:1 ✓	0:0					
5			3:3	2:LEER	1:1	0:0				
6			0:4	3:LEER	2:2	1:1				
7			1:5 ✓	0:LEER	3:3	2:2				
8				3:LEER	2:LEER	1:LEER	0:0			
9					3:LEER	2:LEER	1:1	0:0		
10					0:4	3:3	2:2	1:1		
11					1:5	0:4	3:3	2:2 ✓		
12					2:6	1:5	0:4	3:LEER		
13					3:7	2:6	1:5	0:LEER		
14					0:8 ✓	3:7	2:6	1:LEER		
15						3:LEER	2:LEER	1:LEER	0:0	
16						0:8	3:7	2:LEER	1:1 ✓	
17						1:9	0:LEER	3:LEER	2:LEER	
18						2:10	1:LEER	0:LEER	3:LEER	
19						3:11	2:LEER	1:LEER	0:LEER	
20						0:12	3:LEER	2:LEER	1:LEER	
21						1:13	0:LEER	3:LEER	2:LEER	
22						2:14✓	1:LEER	3:LEER	0:LEER	
23							3:LEER	2:LEER	1:LEER	0:0 ✓

Fig. 10

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

ZEIT	PIXELWERT									
	2	0	5	1	8	15	7	2	1	0
0	0:0									
1	1:1	0:0 ✓								
2	2:2 ✓	1: LEER	0:0							
3	3: LEER	2: LEER	1:1	0:0						
4		3: LEER	2:2	1:1 ✓	0:0					
5			3:3	2: LEER	1:1					
6			0:4	3: LEER	2:2					
7			1:5 ✓	0: LEER	3:3					
8				1: LEER	0:4	3:3	2: WARTEN			
9					1:5	0:4	3: WARTEN	2: WARTEN		
10					2:6	1:5	0:0	1: WARTEN		
11					3:7	2:6	1:1	0:0		
12					0:8 ✓	3:7	2:2	1:1		
13						0:8	3:3	2:2 ✓	1: WARTEN	
14						1:9	0:4	3: LEER	2: WARTEN	
15						2:10	1:5	0: LEER	3: WARTEN	
16						3:11	2:6	1: LEER	0:0	
17						0:12	3:7 ✓	2: LEER	1:1 ✓	
18						1:13	0: LEER	3: LEER	2: LEER	
19						2:14 ✓	1: LEER	0: LEER	3: LEER	
20							3: LEER	2: LEER	1: LEER	0:0 ✓

Fig. 11

Nummer: DE 196 06 178 A1
Int. Cl.[6]: H 04 N 11/02
Offenlegungstag: 26. September 1996

ZEIT	PIXELWERT									
	2	0	5	1	8	15	7	2	1	0
0	0:0									
1	1:1	0:0 ✓								
2	2:2 ✓	1:LEER	0:0							
3	3:LEER	2:LEER	1:1	0:0						
4		3:LEER	2:2	1:1 ✓	0:0					
5			3:3	2:LEER	1:1	0:0				
6			0:4	3:LEER	2:2	1:1				
7			1:5 ✓		3:3	2:2	0:0			
8			2:LEER		0:4	3:3	1:1			
9			3:LEER		1:5	0:4	2:2			
10					2:6	1:5	3:3	0:0		
11					3:7	2:6	0:4	1:1		
12					0:8 ✓	3:7	1:5	2:2 ✓		
13					1:LEER	0:8	2:6	3:LEER		
14					2:LEER	1:9	3:7 ✓		0:0	
15					3:LEER	2:10			1:1 ✓	0:0 ✓
16						3:11			2:LEER	1:LEER
17						0:12			3:LEER	2:LEER
18						1:13				3:LEER
19						2:14 ✓				

Fig. 12